24

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

WINDOWS DNA EN EL DESARROLLO DE UN SISTEMA NERVIOSO DIGITAL

T E S I S

QUE PARA OBTENER EL TITULO DE
INGENIERO EN COMPUTACION
P R E S E N T A N

JOSE LUIS HERNANDEZ GUTIERREZ
VIOLETA HERNANDEZ MARTINEZ
FIDEL SANDOVAL PEREZ

DIRECTOR: ING. GERMAN SANTOS JAIMES

MEXICO, D. F. 2001

Que puedo decir... agradecer a todos aquellos que siempre me dieron el respaldo que siempre necesite, que son tantos y que en este momento al escribir estas líneas, me pone entre la espada y la pared para tener que nombrar a todos en un orden particular ... tengo que ser honesto y liberar mi mente y mi corazón, para dar gracias a mi Dios por permitirme la vida, y a mis padres por haberme criado y educado a sus posibilidades, y que sin ellos no hubiese podido hacer posible este gran logro en mi vida, para compartir con todos mis amigos que siempre me acompañaron, en la Facultad y en donde ahora trabajo. Me hacen falta palabras para expresar el agradecimiento a todos aquellos con los que siempre he contado y a aquellos con los que no. Esta tesis la dedico a todos ustedes ...

A Fidel y a Macaria y a todos sus hijos y sus nietos ...

a los cuates de la Facultad, sin considerar el orden, a Jose Luis, Violeta, al Pollo, Ana, Ramón, Freddy, al Hernán, Germán, Raymundo ...

a los cuates en el trabajo, nuevamente sin considerar el orden, al Trob, al James, a la Patata, a Gaby, al Siquem, Erika, Arturo, al Mac ...

y por ultimo a ti Valeria que cuando leas esta dedicatoria, nuevamente te preguntarás por que no tu ...

Atte.

Fidel Sandoval Pérez

Dedicatorias:

Doy gracias a Dios por brindarme todo lo necesario para poder salir adelante.

Esta tesis y toda mi carrera la dedico a mi Maruchis por todo el apoyo que me ha brindado incondicionalmente a lo largo de toda mi vida, por todo su cariño y sacrificios para que tuviera todo lo necesario y sobre todo por estar ahí, ¡¡¡¡Gracias Mamá!!!!!. (Tarde pero seguro no!!!!!????)

También dedico esta tesis a mis hermanos (Polly Banana y Enano Bodoque), y a toda mi familia que tanto quiero y que tanto me consiente (Papá, Mi Abue Chenta, Lety, Miguelito, Jaime, Manuel, Armando, Gaby, César, Miguel Linares y Tía Flor, Ivan y a todos los demás, que como ustedes saben son muchos muchos y tendría que hacer otra tesis para ponerlos a todos); agradezco de forma especial a gente muy particular en mi vida: Alejandro Velázquez (por todo!!! Gracias Padrino!!!), Ing. Javier Reyes (por su paciencia, consejos y conocimientos), Lucy (por permitirme conocerte), Sr. Carlos (Pollo, por todo su apoyo y tiempo), Cristina Arreola, Sra. Tere (por todas sus recomendaciones) e Ivan Vega (por tus consejos acerca de la vida).

Doy gracias a todos mis profesores, por su tiempo y esfuerzo, a mi querida Universidad por brindarme la oportunidad de ser parte de sus egresados y permitirme formarme como profesionista.

Y por supuesto a todos mis amigos y compañeros: Lizbeth, Marianita, José Luis, Jorgito, Fidel, Marthita, Roberto Cedillo, Andres, Elizabeth, y todos los demás que me han acompañado en diferentes momentos a lo largo de mi vida.

Gracias Josué por compartir tus sueños y tu tiempo conmigo.

Gracias a todos (incluso a los que no recuerdo ahora) ya que de algún modo han contribuido a alcanzar esta meta !!!

Atte.

Violeta Hernández Martínez
Enero 11 '2001

México, D. F. A 11 de Enero de 2001

Deseo dedicar la presente tesis a ustedes mi principal motivo, que en alguna u otra forma me han apoyado para lograr esta meta tan anhelada de una carrera universitaria.

Gracias Madre por ser el mayor de mis apoyos con tu fortaleza, cariño y sacrificios.

A toda mi familia que quiero tanto. A Laura por su cariño, a todos mis grandes amigos Roberto y su Familia, a Eduardo, Rocío, Cristina, Violeta, Manuel, Fidel, Nadia, Ing. Germán, Ing. Luis Marcial Hernández (en memoria de...). Estela, Norma, Alberto, Mauricio y todos los demás que por el momento no me acuerdo, a todos mil gracias.

Muy especialmente quiero agradecer a mi Universidad por permitirme ser orgullosamente parte de ella.

Atte.

José Luis Hernández Gutiérrez

WINDOWS DNA EN EL DESARROLLO DE UN SISTEMA NERVIOSO DIGITAL

TEMARIO

INTRODUCCIÓN

Hoy en día, la tecnología de la información sobre Internet, está permitiendo a las organizaciones reducir dramáticamente sus costos, hacer reingeniería a sus procesos de negocios, y expandir la oferta de sus productos y servicios a sus clientes y socios de negocios. Las compañías que han decidido implantar esta tecnología tienen una ventaja significativa sobre sus competidores, pero esta ventaja, con el acelerado paso del proceso de globalización, esta convirtiéndose en más que una ventaja, una necesidad.

Para sobrevivir en este mundo competitivo y siempre cambiante, es esencial que las organizaciones se encuentren preparadas para los avances tecnológicos y la evolución hacia una economía digital. En tanto más compañías enfilan sus negocios hacia un estilo de trabajo digital, se incrementa la necesidad de compartir información y conocimiento a lo largo de éstas. La expansión del acceso de Internet hacia las redes corporativas internas es de crucial importancia para poder efectuar esta transición de forma exitosa.

Las compañías más exitosas serán aquellas que tengan un acceso rápido a la información y la capacidad de procesar esa información para desarrollar estrategias y adaptarse rápidamente al cambio, ya sea con previa planeación e inclusive sin ésta. La tecnología juega el papel más importante para poder alcanzar estas metas.

El Sistema Nervioso Digital *(Digital Nervous System DNS)* es la visión de Microsoft la cual permite a la tecnología tener un gran impacto sobre los procesos de negocios de las compañías en una economía digital. La información fluye con mayor facilidad en un DNS, permitiendo a las organizaciones manejarse más efectivamente y actuar, reaccionar o adaptarse rápidamente a cualquier imprevisto.

Un DNS se basa en una red de PC´s y software instalado, que permite el flujo rápido y apropiado de la información. Este ayuda a que la toma de decisiones sea rápida y acertada. Prepara a la compañía para reaccionar a eventos no planeados y además ayuda al acercamiento entre la compañía, sus clientes y sus socios de negocios, así también permite a las organizaciones enfocarse directamente en los negocios.

La mayoría de la compañías están optando por Internet y el cómputo distribuido para proporcionar a sus usuarios un acceso rápido a la información. *Windows Distributed InterNet Applications* (Windows DNA) es una plataforma diseñada para construir una nueva generación de soluciones, que contempla el mundo del cómputo personal e Internet. Windows DNA es la primera arquitectura de aplicaciones que abarca e integra por completo a los modelos de desarrollo de aplicaciones Cliente/Servidor y Web.

Utilizando la arquitectura de Windows DNA, los desarrolladores pueden construir aplicaciones de negocios, escalables y multicapa que pueden implementarse sobre cualquier red, proporcionar acceso a diversas fuentes de información aun a través de plataformas distintas y que pueden ser libremente accesadas por cualquier cliente sobre cualquier plataforma. Haciendo uso de la arquitectura Windows DNA, las organizaciones pueden sacar provecho de su tecnología e infraestructura existente y al mismo tiempo adoptar nuevas tecnologías, tales como Internet y la Web para afrontar nuevos requerimientos.

En los próximos años, la tecnología digital incrementará radicalmente la velocidad en que los negocios se llevan a cabo. Cambiará las relaciones entre los proveedores y sus clientes, y transformará los roles individuales de los trabajadores. La facilidad de adaptación que tenga una organización para enfrentar los nuevos retos, dependerá de la forma en que se desarrollen sus procesos digitales internos.

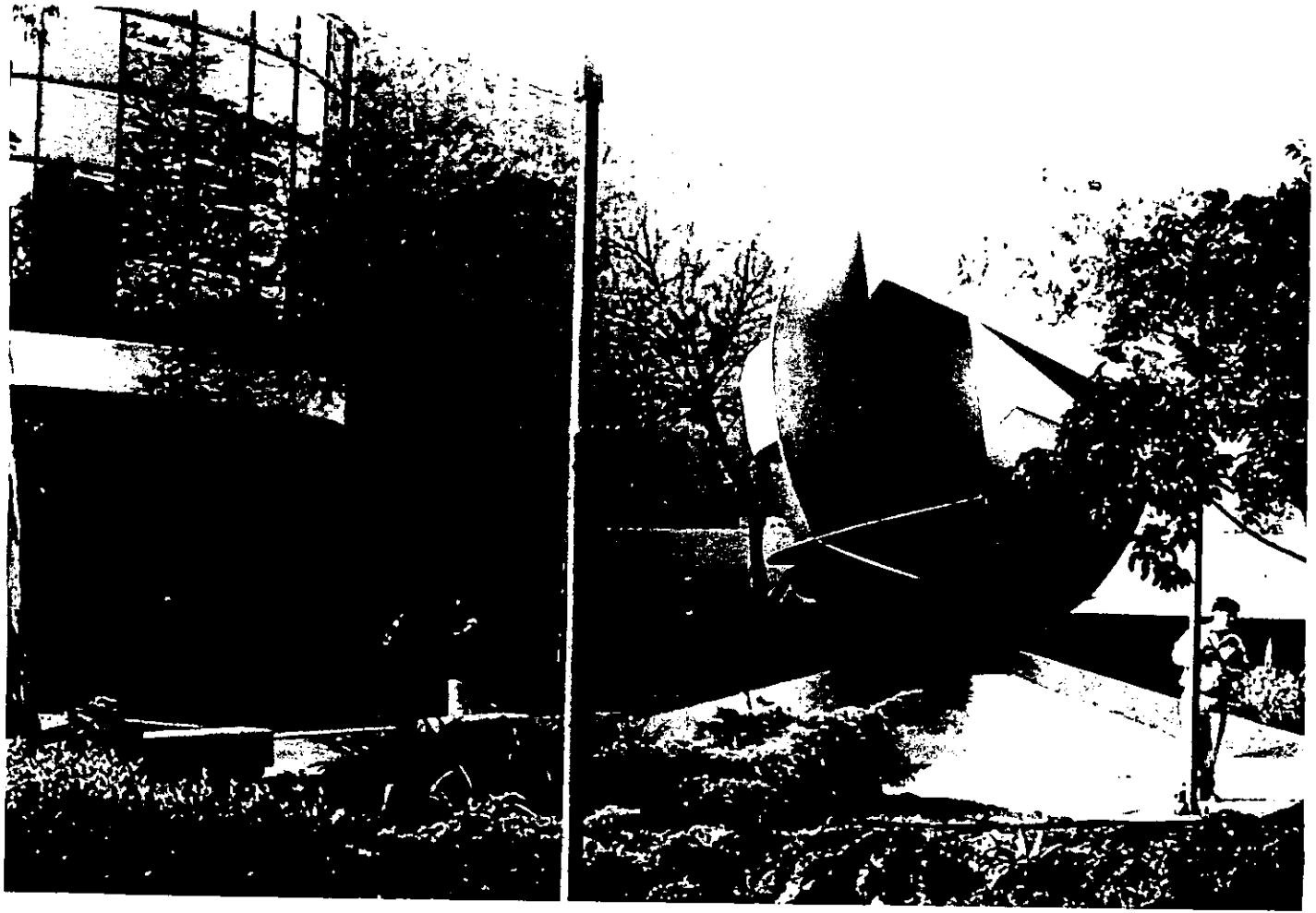
El advenimiento de las nuevas tecnologías, sugiere un nuevo esquema, en el que se integren dichas tecnologías en el desarrollo de la infraestructura del DNS de las organizaciones. Así podemos integrar un Sistema Nervioso Digital, que pueda trabajar en una plataforma distribuida independiente y conformada de tres capas: la capa de datos, la capa de negocios y la capa del cliente, que permita el libre y oportuno flujo de la información haciendo uso de las tecnologías de Internet dentro de una organización.

Las herramientas de desarrollo, así como la infraestructura a elegir, juegan un papel fundamental para el uso efectivo de la tecnología de la información en la construcción de un DNS, de igual forma, permiten a las organizaciones crear, personalizar e integrar aplicaciones corporativas.

La plataforma Microsoft ha sido adoptada por miles de organizaciones, de hecho cada vez más desarrolladores están utilizando estas herramientas, que las de cualquier otro proveedor de software para crear aplicaciones. Miles de aplicaciones de misión crítica han sido desarrolladas utilizando estas herramientas, incluyendo aplicaciones de gran desempeño, desarrolladas por firmas internacionales como United Airlines, NASDAQ, Dell Computer, Solomon Smith Barney, Charles Schwab, Boeing Corporation, Merrill Lynch, y muchas otras.

El desarrollo de esta tesis contempla el uso de herramientas y tecnología de vanguardia, para proporcionar, sin ir demasiado a detalle, una guía en el desarrollo de un Sistema Nervioso Digital haciendo uso de la arquitectura Windows DNA. Es conveniente mencionar que el desarrollo de un sistema involucra diferentes fases que van desde el análisis hasta la implementación del mismo, y que debido a la gran cantidad de conceptos y tecnologías que involucran el desarrollo de un sistema bajo la arquitectura Windows DNA, sólo será posible abarcar parte de estas fases haciendo énfasis en el desarrollo propio del sistema. Así, esta tesis, pretende obtener ventaja de la plataforma y los servicios que ofrece Windows DNA, el manejo y la explotación de la información que otorga un Sistema Nervioso Digital en una esquema que englobe estas tecnologías, y que ofrezca ventajas considerables sobre el esquema convencional de dos capas (cliente/servidor) y que vaya más allá del propio esquema de tres capas (datos, negocios y cliente), obteniendo con todo esto la puesta en práctica de lo mejor de cada una de las tecnologías en el desarrollo de un Sistema Nervioso Digital elemental que permita dar soluciones reales y oportunas sobre el flujo de la información, así como ocasionar una revolución en el manejo del conocimiento dentro de una organización.

Lo anterior sugiere uso de tecnología accesible y de vanguardia y no el manejo de una plataforma específica para el diseño y desarrollo de un DNS que aborde las necesidades elementales de una organización.



CAPÍTULO UNO

SISTEMA NERVIOSO DIGITAL

En la actualidad, la forma en que las compañías o empresas usan y manejan la información puede ser un factor determinante para el éxito o fracaso de las mismas. Y la manera en que se usa y maneja la información depende directamente de la tecnología que se utiliza.

El ritmo al que evoluciona y avanza la tecnología de la información, hará que en los próximos años la tecnología digital incremente notablemente su influencia en los negocios y las empresas, cambiando las relaciones de las empresas con sus clientes y transformando el papel de los empleados dentro de las mismas. Por lo tanto, en el grado en que los procesos digitales de una empresa se adecuen a las necesidades de la misma, dependerá la adaptación de ésta, al nuevo y cambiante panorama empresarial.

DEFINICIÓN

Un Sistema Nervioso Digital *(Digital Nervous System DNS)* es un sistema de trabajo que representa una nueva y eficaz forma de administrar los negocios. El DNS ayuda a las compañías a tomar mejores decisiones, lo que les permite ahorrar tiempo y dinero, operar sus negocios de manera más eficaz y establecer relaciones más sólidas con sus clientes y asociados, de una manera muy semejante a aquella en que el sistema nervioso biológico ayuda al cuerpo humano a sobrevivir.

Un DNS se construye a base de una combinación de hardware y software, no se trata de un producto sino de un conjunto de sistemas, infraestructura y tecnologías combinadas en una forma de trabajo. Pensemos en el sistema nervioso biológico del ser humano, este controla los sistemas básicos (respiración, circulación sanguínea, aparato digestivo, etc.) que hacen posible la vida, recibe además los estímulos sensitivos, los transmite al cerebro y elige instantáneamente la respuesta más adecuada. El DNS realiza funciones paralelas dentro de una organización. Permite que los procesos internos de la compañía operen rápida y eficazmente, de forma que la organización pueda responder al cliente de forma inmediata y reaccione a las condiciones cambiantes del mercado competitivo, ofreciendo a los empleados todo el poder de la información de la corporación.

La clave radica en cuán efectivamente la organización gestiona el flujo de su información. Ahora cualquier tipo de información (números, texto, sonido, vídeo) puede ser adecuado al formato digital. La amplia disponibilidad de hardware y software ha hecho posible y necesario el redimensionamiento del modo de hacer negocios para todo tipo de empresas. De hecho, muchas compañías ya han realizado las inversiones necesarias en tecnologías de información para permitir a sus empleados obtener, almacenar, compartir y utilizar la información de una forma nueva y más competitiva.

Por lo tanto, la idea principal consiste en emplear la tecnología de forma que mejore la manera de trabajar de la gente y transforme el sistema de operación de la compañía. Las empresas deben crear procesos digitales que gestionen y distribuyan toda la información importante que los empleados necesiten, ayudándoles a responder de forma efectiva a los

cambios y nuevas oportunidades, acelerando su tiempo de respuesta y permitiéndoles solucionar todo tipo de problemas en el mismo momento en que éstos se susciten.

No existe un sistema nervioso digital estándar. Puesto que cada empresa es diferente, sus sistemas digitales deberán reflejar las particularidades de su estructura y necesidades concretas.

Existen tres principios que permite a una compañía aprovechar su infraestructura tecnológica existente y crear sistemas que permitan a los empleados utilizar todo el conocimiento almacenado dentro de la organización, lo cual permite responder con rapidez y creatividad a las exigencias del mercado.

Estos principios son:

- **Administración del Conocimiento.**

  Combinar la experiencia personal con los datos organizacionales para tomar mejores decisiones, mejorando el flujo de la información y las relaciones entre los trabajadores de la compañía. El DNS, ofrece a los usuarios opciones que los ayudan a colaborar, analizar, seguir y publicar la información de modo más eficaz.

- **Operaciones de Negocios.**

  Ahorrar tiempo y dinero, operar los negocios en una forma que resulte más eficiente. El DNS permite que una organización automatice sus operaciones de negocios, de modo que todos los integrantes de la empresa dispongan de datos absolutamente actualizados.

- **Comercio.**

  Consolidar relaciones con clientes y asociados. El DNS ofrece a las compañías la oportunidad de utilizar Internet para establecer relaciones más fuertes con sus clientes y asociados, debido a que en la actualidad prácticamente cualquier empresa esta conectada a Internet.

BENEFICIOS DE UN SISTEMA NERVIOSO DIGITAL

Un DNS puede traer grandes beneficios a las compañías que cuenten con él, como la habilidad de pensar y actuar más rápidamente. Así como la habilidad de conocer mejor a sus clientes.

Un DNS brinda una ventaja competitiva ya que permite:

- **Actuar rápidamente.**

  Se puede lograr que todos los colaboradores trabajen como si fueran uno solo. Pues al optimizar sus procesos de manejo de información a través de procesos digitales, se eliminaran tiempos muertos.

- **Reaccionar a imprevistos**.

  Un DNS prepara a las compañías para los imprevistos. Les proporciona toda la información necesaria para responder inteligentemente a situaciones inesperadas. También permite reestructurar rápidamente prioridades, lograr afrontar las amenazas y posiblemente obtener ventaja de la situación.

- **Tomar las decisiones comerciales con la información más reciente.**

  Un DNS asegura que usted puede contar en el instante con la información más reciente y exacta, en cualquier momento y en cualquier lugar.

- **Estar más cerca de los clientes**

  Una compañía que cuenta con un DNS da respuesta inmediata y oportuna a las necesidades de sus clientes, lo cual ayuda a fortalecer las relaciones entre clientes y proveedores.

- **Enfocarse en el negocio y no en la tecnología**

  Un DNS es fácil de usar porque esencialmente es software que trabaja en conjunto de forma consistente, de esta manera no se tiene que invertir tiempo en averiguar como es que los procesos trabajan. Las compañías se pueden enfocar más en los resultados que en las herramientas utilizadas para obtenerlos.

COMERCIO Y EL SISTEMA NERVIOSO DIGITAL

Forjar buenas relaciones con clientes y socios comerciales es la clave para tener éxito en los negocios. El surgimiento del comercio electrónico ofrece la oportunidad perfecta para las empresas de construir y fortalecer estas relaciones gracias a Internet, el canal de comunicaciones perfecto (rápido, seguro, barato y muy accesible).

Empresas de todos los tamaños podrán notar los beneficios de desarrollar una estrategia de comercio en línea, pues serán estas compañías las mejor preparadas para satisfacer las demandas de la Era Digital y conseguir así una ventaja competitiva sobre las demás.

Algunos beneficios son obvios: costos bajos, mayor alcance, menor tiempo de respuesta, etc. Pero el comercio electrónico no es sólo poner productos a la venta más rápidamente, Internet también profundizará sus relaciones con clientes y socios.

Gracias a las comunicaciones y transacciones digitales las compañías pueden ofrecer (y los consumidores pueden esperar) mejores servicios y soporte. De hecho, las ventas en línea se están convirtiendo en una parte fundamental de las empresas, pues el autoservicio y el acceso rápido a grandes cantidades de información en la Web (incluyendo los productos de la competencia) da a los consumidores mayor influencia y poder.

La tendencia a futuro, es que todas las compañías usen Internet para fortalecer sus relaciones con clientes y socios. La tecnología, el mercado y las alianzas con proveedores

de soluciones serán piedra angular para construir soluciones comerciales de éxito como parte integral de un DNS.

Estas soluciones suelen referirse a cuatro tipos de procesos empresariales:

1. **Ventas, Servicios y Marketing Directo.**

   En el cual se incluyen el desarrollo de marcas, la venta directa y el servicio de atención al cliente, tanto de cara a las relaciones entre empresas como a las relaciones entre las empresas y sus clientes.

2. **Servicios financieros y de información.**

   La facturación en línea, los servicios de inversión, los servicios bancarios a domicilio y la distribución de bienes y contenidos digitales, se encuentran comprendidos en este rubro.

3. **Compras corporativas.**

   Internet puede ayudar a automatizar los procesos manuales de la mayoría de las compañías, rediseñando los procesos de compra mediante una aplicación de autoservicio para los compradores y una de intercambio para los proveedores.

4. **Cadena de valor**

   El comercio vía Internet fortalece las relaciones entre las empresas, creando una cadena de valor mas dinámica para reducir los requerimientos de inventario, acortar los ciclos de facturación e incrementar la capacidad de respuesta de las empresas ante las necesidades de sus clientes.

REQUERIMIENTOS PARA LA INSTALACIÓN DE UN SISTEMA NERVIOSO DIGITAL

Un Sistema Nervioso Digital se construye tomando en cuenta seis puntos principales:

1. **Contar con una arquitectura de cómputo**

   La flexibilidad de una arquitectura de cómputo permite a las compañías adaptarse de forma rápida y accesible al crecimiento y a los cambios. La PC proporciona una combinación optima de flexibilidad, precio y rendimiento para la mayoría de las organizaciones, y por esto proporciona la base fundamental que hace posible un DNS.

2. **Toda la información debe estar en formato digital**

   Una premisa fundamental de un DNS es que toda la información debe estar disponible en un formato digital. La información digital es más fácil de crear y mantener, asi también puede ser analizada, se pueden ejecutar búsquedas sobre esta, puede ser actualizada con facilidad y puede compartirse ampliamente.

3. **Contar con correo electrónico**

   El correo electrónico permite a los miembros de un equipo responder rápidamente a las preguntas que surgen día a día. Además también facilita la construcción de sistemas de flujo de trabajo (WorkFlow) que están completamente integrados con el resto de los procesos computacionales en el resto de la compañía.

4. **Conectividad en ambas direcciones.**

   Además de contar con un correo electrónico se requiere de una red que permita a la compañía conectar todos sus sistemas de computo y de Internet. El software que soporte a un DNS debe funcionar correctamente sobre Internet de tal manera que las compañías puedan extender sus sistemas computacionales para tomar ventaja de Internet.

5. **Las herramientas de productividad deben ser comunes para toda la compañía**

   Esto permite a cada uno de los miembros dentro de la compañía usar e intercambiar información de una forma predecible y consistente, por lo cual resulta razonable esperar que nuevos empleados sepan como utilizar los sistemas existentes.

6. **Aplicaciones especificas de negocios**

   Todos los negocios necesitan de aplicaciones especializadas para ayudarse a llevar a cabo procesos específicos. Estas aplicaciones ofrecen el acceso a la información a más individuos dentro de la organización para poder tomar mejores decisiones de negocios.

CONSTRUCCIÓN DE UN SISTEMA NERVIOSO DIGITAL

Como ya se mencionó no existe una forma estándar de construir un sistema nervioso digital. Los recursos, la estructura y las necesidades de la compañía determinarán las soluciones que sean necesarias y la manera de implementarlas.

Un DNS se encuentra conformado por diferentes componentes. La siguiente tabla muestra las cinco partes que constituyen a un DNS además de una breve descripción de cada una de ellas.

COMPONENTE	DESCRIPCIÓN
Inteligencia de Negocios	Una base de conocimiento construida a partir de datos operacionales
Comercio Electrónico	Múltiples líneas de aplicaciones de negocios formando una línea que mantiene juntos a los negocios a través de Internet

Línea de Negocios	Procesamiento de órdenes y otros métodos de negocio incorporando, por ejemplo MS SQL Server y COM+
Colaboración	Integración en equipo alrededor de los procesos de negocio.
Seguimiento	Una vista integrada del proceso de negocio en su totalidad, desde un número determinado de aplicaciones poderosas front-end.

La primera de las cinco partes del DNS es la *Inteligencia de Negocios.*

Este componente puede ser considerado como el cerebro del sistema, dado que aquí es donde las experiencias y el conocimiento de la compañía se encuentran almacenados en un almacén de datos, permitiendo el análisis del negocio con herramientas para toma de decisiones. Aquí los éxitos y fracasos de una compañía se encuentran almacenados.

El siguiente componente es el *Comercio Electrónico.*

Esta parte habilita a las compañías para conectarse con sus clientes y proveedores. Esta puede considerarse la apariencia externa del DNS si hacemos una analogía con la piel en el sistema nervioso biológico del cuerpo humano. Con el uso de tecnologías cliente/servidor el componente de comercio electrónico reacciona a las influencias externas de la compañía.

El corazón y el alma del sistema, la *Línea de Negocios.*

En esta parte es donde las órdenes son procesadas y la base de datos de clientes es alimentada. Tecnologías como SQL, Windows 2000, COM+, Transaction Server, y ASP juegan un papel vital en el futuro de este componente. Microsoft visualiza al DNS con la capacidad de integración con sistemas como UNIX y ambientes mainframe.

El componente de *Colaboración*.

Esta constituido de herramientas y tecnologías para integrar los equipos de negocios. En esta parte las intranets corporativas y las infraestructuras de mensajería como e-mail y servidores de noticias son desarrolladas, implementadas y mantenidas. Herramientas como MS Exchange y Outlook, en conjunto con los servidores de Web y los clientes son utilizados para tomar ventaja del conocimiento de la compañía.

El componente de *Seguimiento*.

Como los ojos y los oídos, observa y escucha el flujo del negocio. Los clientes Web son usados para monitorear las aplicaciones que siguen los procesos de negocios. Es aquí donde eventos inesperados como el decremento en las ventas o el incremento de quejas recibidas, son fácilmente observados.

EL FUTURO

La Era Digital hoy en día está causando profundas perturbaciones en numerosas empresas. Tomemos por ejemplo lo que esta ocurriendo en la industria de los libros y la música, las ventas informáticas y el material de oficina. Han aparecido nuevas empresas y nuevos líderes, y los que deseen sobrevivir en este contexto deberán ajustar dramáticamente sus modelos empresariales. Estos cambios seguirán acrecentándose en los próximos años, en tanto un número creciente de gente se vaya conectando a la Web y acomodándose a las transacciones digitales, y en tanto las compañías encuentren mejores maneras de llevar sus negocios a la Web (personalizando sus sitios Web, etc.)

Las empresas que se ajusten a estas nuevas circunstancias y desarrollen un sistema nervioso digital de forma eficaz e imaginativa podrán sobresalir.

En último término, un DNS transformará los tres elementos principales de toda empresa:

1. Las relaciones con sus clientes y socios comerciales (comercio)

2. El flujo de la información y las relaciones entre los trabajadores de la compañía (gestión del conocimiento)

3. Los procesos empresariales internos (operaciones empresariales).

CAPÍTULO DOS

WINDOWS DNA

Hoy en día, la convergencia de Internet y las tecnologías de cómputo de Windows promete oportunidades nuevas y excitantes en los negocios: para crear una nueva generación de soluciones de cómputo que mejore dramáticamente la respuesta de una organización, para utilizar Internet y el Web de manera más efectiva para alcanzar a los clientes directamente, y para dar mejor acceso a la información en cualquier momento y en cualquier lugar. Cuando un sistema tecnológico ofrece éstos resultados, entonces es llamado Sistema Nervioso Digital.

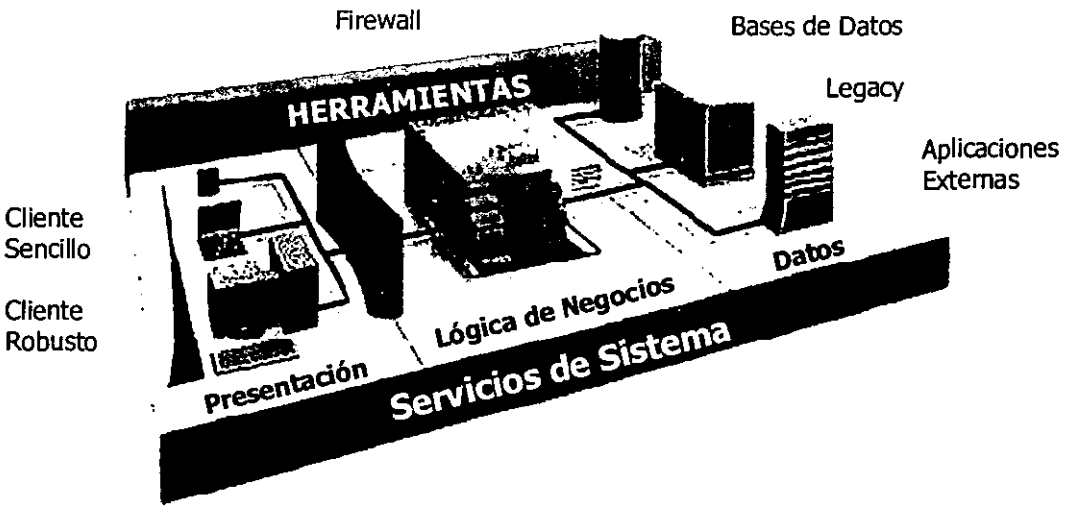
La creación de un verdadero Sistema Nervioso Digital requiere de compromiso, tiempo e imaginación. No es algo que cualquier compañía tenga la determinación de llevar a cabo, pero aquellos que lo logren tendrán una ventaja importante sobre aquellos que no. En la creación de un Sistema Nervioso Digital, las organizaciones se enfrentan a muchos retos, que podemos resumir en las siguientes preguntas:

- ✓ ¿Cómo tomar ventaja de las nuevas tecnologías de Internet y preservar al mismo tiempo las inversiones en gente, aplicaciones y datos?
- ✓ ¿Cómo construir una solución moderna, escalable y que sea dinámica y flexible al cambio?
- ✓ ¿Cómo disminuir el costo total en materia de cómputo y al mismo tiempo poner a trabajar ambientes de cómputo complejos?

INTRODUCCIÓN A WINDOWS DNA

Windows DNA (*Windows Distributed interNet Applications Architecture*) es el esquema de trabajo de Microsoft para la construcción de una nueva generación de soluciones de negocios altamente adaptables que permite a las compañías explotar al 100% los beneficios del Sistema Nervioso Digital.

Windows DNA es la primera arquitectura de aplicaciones que abarca y se integra completamente con Internet, Cliente/Servidor y los modelos de cómputo de PC para una nueva clase de soluciones de cómputo distribuidas.



Utilizando el modelo Windows DNA los clientes pueden construir aplicaciones de negocios modernas, escalables y multicapa que pueden funcionar en cualquier red. Las aplicaciones Windows DNA pueden mejorar el flujo de información dentro y fuera de una organización, son dinámicas y flexibles a las necesidades de cambio en los negocios, y pueden ser fácilmente integradas con los datos y los sistemas existentes. Debido a que las aplicaciones Windows DNA requieren de que la plataforma de los servicios de Windows trabajen en conjunto, las organizaciones pueden entonces enfocarse en ofrecer soluciones de negocios en lugar de preocuparse en la integración de la plataforma y de sus servicios.

LOS PRINCIPIOS DE WINDOWS DNA

Los siguientes principios fueron la guía para el desarrollo de la Arquitectura Windows DNA:

El Web sin compromiso. Las organizaciones quieren crear soluciones que exploten completamente el alcance y las capacidades de comunicación de Internet, mientras habilitan a los usuarios finales con la flexibilidad y el control de las aplicaciones de PC de hoy en día. En resumen, requieren de tomar ventaja sobre Internet, sin tener que comprometer su habilidad para explotar los avances en tecnología de PC.

Interoperabilidad. Las organizaciones quieren que las nuevas aplicaciones que construyan trabajen en conjunto con sus aplicaciones existentes y expandir éstas aplicaciones con nueva funcionalidad. Requieren soluciones que se apeguen a protocolos y estándares abiertos de manera que otros proveedores de soluciones se puedan integrar

fácilmente. Desaprueban ser forzados a reescribir aplicaciones legendarias que aún se encuentran en uso y miles que aún se encuentran en desarrollo.

Integración Verdadera. Para que las organizaciones implementen aplicaciones distribuidas verdaderamente escalables y manejables satisfactoriamente, puntos clave como seguridad, manejo y monitoreo de transacciones, servicios de componentes y servicios de directorio deben ser desarrollados, probados y proporcionados como características integrales de la plataforma. En muchas otras plataformas éstos servicios críticos son proporcionados por separado, las soluciones no integradas de diferentes proveedores forzan a los profesionales a funcionar como integradores de sistemas.

Bajo costo de propiedad. Las organizaciones quieren proveer a sus clientes con soluciones fáciles de implementar y manejar, y fáciles de cambiar y evolucionar con el tiempo. Requieren de soluciones que no involucren un esfuerzo intensivo y gran cantidad de recursos para poderse implementar en un ambiente de trabajo, y reducir el costo de propiedad tanto en el lado del servidor como en el cliente.

Puesta rápida en el mercado. Las organizaciones quieren ser capaces de poder llevar a cabo todo lo anterior, mientras se apegan a los calendarios de los proyectos, utilizando herramientas de desarrollo comunes y sin la necesidad de una reducación masiva o un cambio paradigmático en la forma en que desarrollan software.

TECNOLOGÍAS WINDOWS DNA

Los requerimientos de todas las capas de aplicaciones distribuidas modernas: interface de usuario y navegación, procesos de Negocios y almacenamiento, se encuentran comprendidas en un conjunto bien constituido denominado Windows DNA *plataform tecnologies and services.*

El corazón de Windows DNA es la integración del Web y el modelo de desarrollo de aplicaciones Cliente/Servidor a través del *Component Object Model* (COM) Los servicios de Windows DNA son expuestos en una forma unificada a través de COM para que las aplicaciones los usen. Éstos servicios incluyen manejo de componentes, HTML dinámico, *Web browser* y *Web server, scripting*, transacciones, *message queuing*, seguridad, directorio, acceso a datos y a bases de datos, manejo del sistema e interface de usuario.

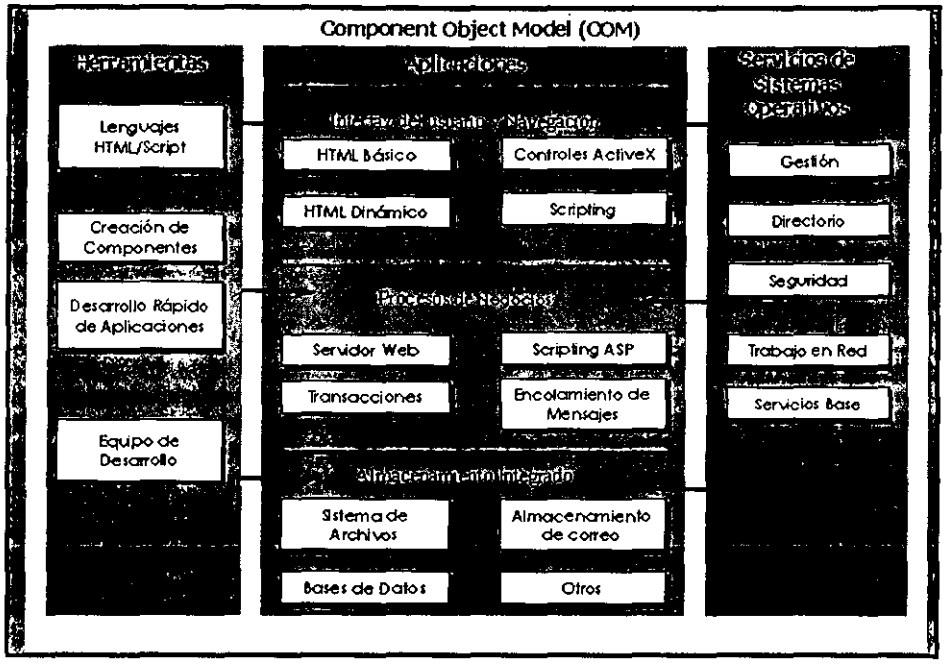
Windows DNA ésta basado en muchos estándares importantes aprobados por organizaciones tales como el *World Wide Web Consortium* (W3C) (http://www.w3c.org/) y el Internet *Engineering Task Force* (IETF) (http://www.ietf.org/). Apegándose a protocolos e interfaces publicadas se facilita la integración con otras soluciones y proporciona una gran interoperabilidad con los sistemas existentes.

Debido a que Windows DNA ésta basado en COM y estándares abiertos de Internet, los desarrolladores pueden utilizar cualquier lenguaje o herramienta para crear aplicaciones compatibles. COM proporciona un modelo de objetos moderno e independiente del lenguaje que proporciona una forma estándar para que las aplicaciones interoperen en todas las capas de la arquitectura. Por medio de COM los desarrolladores pueden extender cualquier parte de la aplicación a través de componentes de software que pueden ser desarrollados en C++, Visual Basic, Java u otro lenguaje. Debido a ésta apertura en la arquitectura, Windows DNA soporta una gran cantidad de herramientas de

desarrollo, incluyendo herramientas de Borland, ProwerSoft, Microsoft y muchos otros proveedores.

FAMILIA DE TECNOLOGÍAS DE WINDOWS DNA

Windows Distributed interNet Architecture (Windows DNA) unifica los servicios disponibles en las computadoras personales, servidores de aplicación y mainframes, el diagrama siguiente sumariza las categorías de tecnologías que constituyen a Windows DNA.

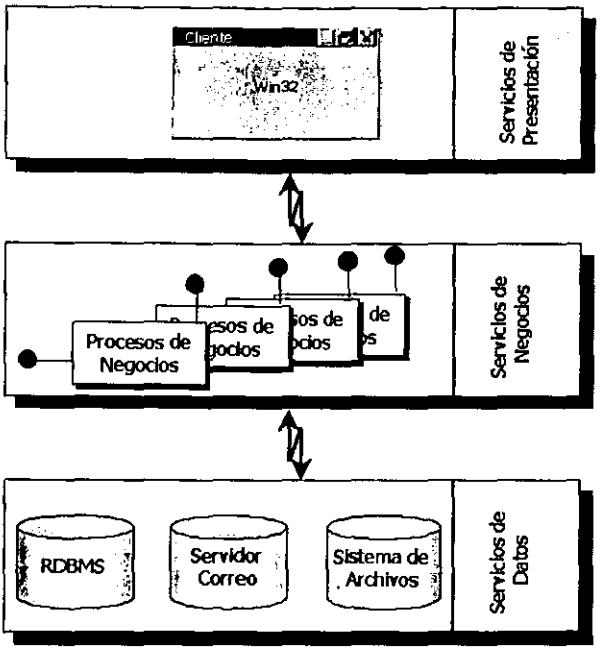


Diseño y construcción de aplicaciones Windows DNA

Para diseñar y construir aplicaciones en tres capas (*three-tiered*) sobre la plataforma de Windows, los desarrolladores deben comprender dos cosas: los fundamentos del diseño en general de una aplicación en tres capas y tecnologías especificas de Microsoft que son relevantes en el desarrollo de aplicaciones en tres capas sobre la plataforma de Windows. Windows DNA describe ambas.

Windows DNA es una arquitectura general que describe como construir aplicaciones en tres capas para la plataforma Windows. El propósito de éste capítulo es proporcionar a los desarrolladores una metodología general para el diseño y desarrollo de aplicaciones Windows DNA. Éste capitulo no pretende describir cada aspecto concebible de Windows

DNA, más bien enfocarse en proporcionar un firme entendimiento de los principios básicos del diseño de una aplicación Windows DNA.



Una aplicación en tres capas es una aplicación cuya funcionalidad puede ser segmentada en tres capas lógicas de funcionalidad: servicios de presentación, servicios de negocios y servicios de datos.

La capa de servicios de presentación es responsable de:

- ✓ Recolectar la información proveniente del usuario
- ✓ Mandar la información recolectada del usuario hacia la capa de servicios de negocios para su procesamiento.
- ✓ Recibir los resultados de la capa de servicios de negocios.
- ✓ Presentar los resultados al usuario.

La capa de servicios de negocios es responsable de:

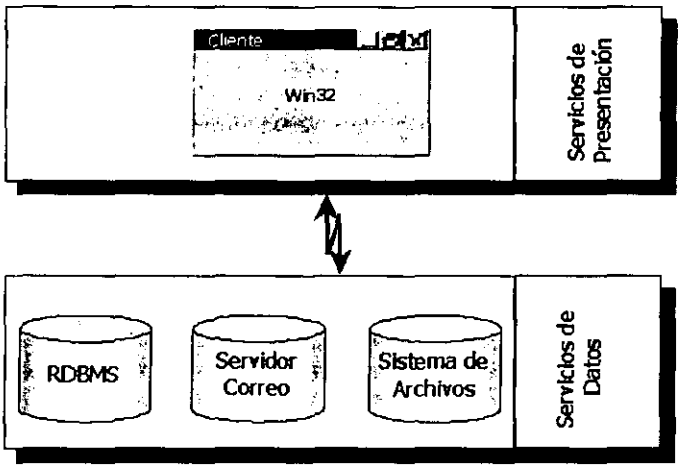
- ✓ Recibir la entrada de la capa de presentación.
- ✓ Interactuar con los servicios de datos para llevar a cabo las operaciones de negocios las cuales la aplicación pretende automatizar (por ejemplo, calcular un impuesto, procesamiento de ordenes de compra, etc.)
- ✓ Mandar los resultados procesados a la capa de presentación.

La capa de servicios de datos es responsable de:

- ✓ Almacenamiento de los datos.
- ✓ Recuperación de los datos.
- ✓ Mantenimiento de los datos.
- ✓ Integridad de los datos.

Los servicios de datos los podemos encontrar en una gran variedad de formas y tamaños, incluyendo los Manejadores de Bases de Datos Relacionales (RDBMS's) como SQL Server, servidores e-mail como Exchange Server, y los sistemas de archivos como NTFS.

Si comparamos, una aplicación en dos capas (*two-tiered*) es una aplicación cuya funcionalidad puede ser segmentada en dos capas lógicas, los servicios de presentación y los servicios de datos, y mientras las responsabilidades de la capa de datos son las mismas para una aplicación de dos capas que para una aplicación de tres capas, las responsabilidades para los servicios de presentación no son las mismas. En una aplicación de tres capas los servicios de presentación son responsables de recolectar la información del usuario, mandar ésta información hacia los servicios de negocios para su procesamiento, recibir los resultados de los servicios de negocios y presentar esos resultados al usuario. En contraste los servicios de presentación de una aplicación de dos capas son un tanto diferentes. Los servicios de presentación de una aplicación de dos capas son responsables de recolectar la información del usuario, interactuar con los servicios de datos para llevar a cabo las operaciones de negocios de la aplicación, y presentar los resultados de esas operaciones al usuario.



Una aplicación de dos capas es una aplicación cuya funcionalidad puede ser segmentada solo en dos capas: servicios de presentación y servicios de datos.

La diferencia en responsabilidades de los servicios de presentación para las aplicaciones de dos y tres capas influyen enormemente sobre los desarrolladores en la perspectiva de una aplicación como tal. Por ejemplo, en un diseño de dos capas la lógica que lleva a cabo las tareas que una aplicación pretende automatizar (por ejemplo, calcular un impuesto, procesamiento de órdenes de compra, etc.), la cual es normalmente concebida como "la aplicación", se ejecuta en donde los servicios de presentación estén localizados.

Sin embargo, en un diseño de tres capas "la aplicación" se ejecuta donde los servicios de negocios están localizados. Tales diferencias fundamentales, pueden llevar a todo tipo de confusiones cuando se discutan conceptos de aplicaciones de tres y dos capas, por esto es crucial entender la diferencia fundamental en la perspectiva del desarrollador asociada al diseño de aplicaciones de dos y tres capas.

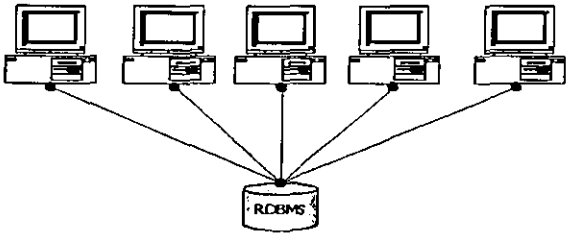
OBJETIVOS DE DISEÑO DE WINDOWS DNA

La arquitectura Windows DNA está diseñada para maximizar el desempeño de una aplicación:

Autonomía

La autonomía de una aplicación se refiere a la habilidad de una aplicación para administrar sus recursos críticos. Los recursos críticos son los recursos que se requieren para que una aplicación funcione confiablemente como una entidad independiente, conexiones con RDBMS's, conexiones con mainframes, y transacciones, son ejemplos de recursos críticos. La autonomía de una aplicación es uno de los aspectos más importantes del diseño de una aplicación Windows DNA, y también es una de las diferencias más importantes entre los diseños cliente/servidor de dos y tres capas.

En un diseño Cliente/Servidor típico de dos capas, los clientes tienen acceso directo a los recursos críticos de la aplicación y son libres de utilizar dichos recursos en el momento que lo deseen.



Las aplicaciones Cliente/Servidor de dos capas son no autónomas, lo que significa que los clientes tienen acceso directo a los recursos críticos de la aplicación y son libres de utilizarlos en el momento que lo deseen.

Debido a que los clientes tiene acceso directo a los recursos críticos de la aplicación, existe muy poco o casi nada que una aplicación pueda hacer para protegerse de conductas maliciosas o inesperadas, las cuales al final comprometen la estabilidad global de la aplicación. Por ejemplo, un cliente mal intencionado podría proponerse agotar los recursos críticos de la aplicación en el afán de evitar que los otros clientes lleven a cabo su trabajo. Un ataque como éste podría dejar inutilizable a la indefensa aplicación.

Las aplicaciones Windows DNA, por otra parte, nunca permiten a los clientes tener acceso directo a los recursos críticos. En lugar de eso, los clientes mandan sus peticiones a componentes de confianza llamados ejecutantes (*executants*) que llevan a cabo las operaciones de negocios que la aplicación pretende automatizar (por ejemplo, calcular un impuesto, procesamiento de órdenes de compra, etc.) Digamos que OrdenCompra es un ejecutante que podría llevar a cabo las operaciones necesarias para adicionar una línea

de producto a una orden de compra, cerciorarse que el producto deseado se encuentre en existencia, calcular el IVA, cargos por envío, etc. Al forzar a los clientes para que manden sus peticiones a los ejecutantes y a su vez éstos lleven a cabo las operaciones de negocios utilizando los recursos críticos de forma confiable y bien definida, las aplicaciones Windows DNA quedan en completo control de sus recursos críticos, lo que finalmente incrementa la estabilidad global de la aplicación. Debido a que los ejecutantes son componentes confiables éstos tienen acceso directo a los recursos críticos, lo que significa que deben de poner atención especial en la forma en la cual los recursos críticos son utilizados. Antes de que un ejecutante lleve a cabo cualquier operación de negocio en nombre de un cliente, éste debe autentificar la identidad del cliente, validar que el cliente tenga la autorización apropiada para llevar a cabo la operación de negocios solicitada, y validar los datos y que la petición del cliente sea sintácticamente correcta. Cualquier cosa que no sea una petición con un formato perfecto debe ser rechazada inmediatamente Con tal estándar de excelencia, es razonable el esperar un gran número de peticiones rechazadas, por esto las aplicaciones Windows DNA debe proporcionar ejecutantes emisarios, emisarios (*emissaries*)

Los emisarios son componentes que está diseñados para ayudar a los clientes para enviar sus peticiones en el formato correcto a los ejecutantes para llevar a cabo las operaciones de negocios en su nombre. Una de las formas en que los emisarios ayudan a los clientes es en proporcionarles cualquier dato suplementario que el cliente pudiese necesitar para preparar su petición. Éste tipo de información pueden ser listas de sólo lectura que son mantenidas por el ejecutante (listas de precios, números de identificación de inventario, etc.), o datos gravables por el usuario que son mantenidos por el emisario en nombre del cliente (información del perfil de usuario, contenido de una *shopping cart*, etc.) Otra forma en que los emisarios ayudan a los clientes es validando tanto como sea posible la información proporcionada por el usuario antes de transmitir la petición al ejecutante, lo cual ayuda a minimizar él trafico de red. Para reducir aún más el tráfico de red, los emisarios, y cualquier dato suplemental que puedan requerir, pueden ser descargados y ejecutados directamente en el cliente. Cuando se transmiten las peticiones al ejecutante, los emisarios son libres de utilizar cualquier medio a su disposición (por ejemplo, DCOM, MSMQ, Winsock, etc.)

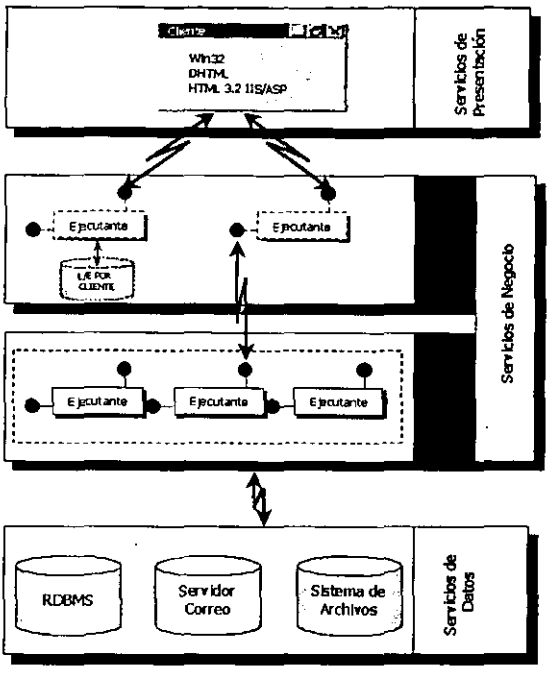
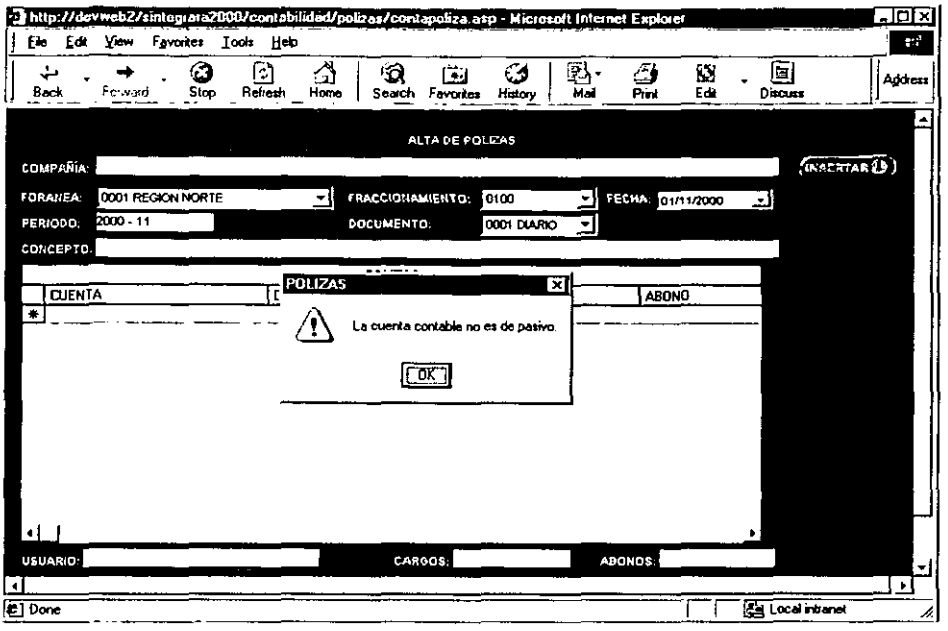


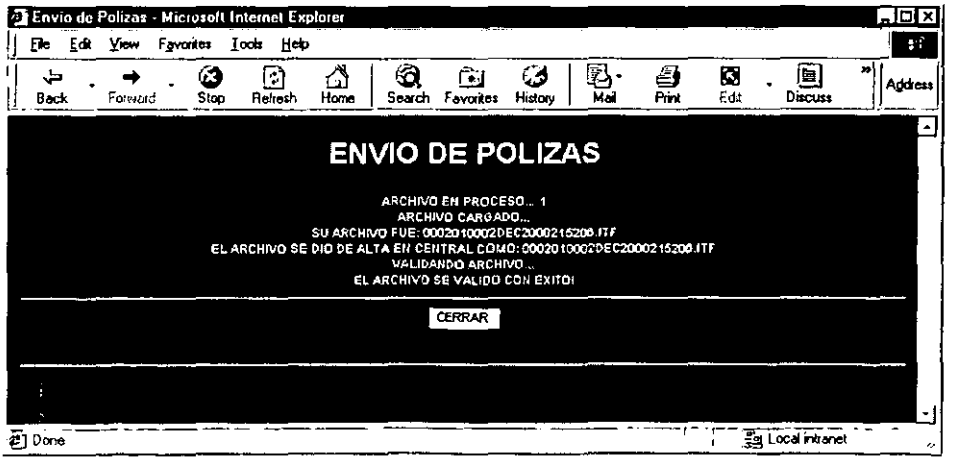
Diagrama de arquitectura de varios componentes involucrados en la construcción de una aplicación Windows DNA utilizando la metodología de ejecutantes y emisarios.

Para que una aplicación Windows DNA pueda hacer que sus operaciones de negocios se encuentren disponibles al más amplio rango de servicios de presentación (HTML dinámico, Microsoft Win32, y HTML), los emisarios deben ser accesibles desde servicios de presentación interactivos y no interactivos. Los servicios de presentación interactivos son capaces de proporcionar retroalimentación al usuario ya sea en una base form-by-form o field-by-field. Por ejemplo, una aplicación que soporta una interfaz de usuario Win32 puede desplegar inmediatamente un mensaje de error en respuesta a la entrada de un dato no válido en el campo de una forma, ésta capacidad de validación antes era exclusiva de las interfaces Win32. Hoy en día es posible implementar éstas capacidades a interfaces HTML.



Los servicios de presentación interactivos como los creados utilizando Win32 ó DHTML son capaces de proporcionar retroalimentación inmediata al usuario en una base form-by-form o field-by-field.

Los servicios de presentación no interactivos, por otro lado, son capaces de proporcionar retroalimentación al usuario sólo en una base form-by-form. Por ejemplo, una aplicación que soporta una interfaz de usuario HTML simple requiere que el usuario envíe la forma completa al servidor antes de que llevar a cabo cualquier tipo de procesamiento.



Los servicios de presentación no interactivos como los creados en HTML simple no son capaces de proporcionar una retroalimentación inmediata al usuario, porque requieren de una consulta al servidor para llevar a cabo cualquier y toda rutina de validación de datos.

Debido a que los servicios de presentación no interactivos requieren de una consulta al servidor para llevar a cabo sus rutinas de validación, están limitados a proporcionar retroalimentación al usuario en una base form-by-form.

Mientras la interfaz de usuario HTML puede ser extremadamente no interactiva, es bien soportada por una amplia variedad de plataformas diferentes a Windows (Non-Windows), incluyendo Linux, UNIX, y Macintosh, lo que hace que a HTML ideal para extender el alcance de una aplicación hacia una gran variedad de plataformas. Las aplicaciones Windows DNA deben ser diseñadas para dar cabida tanto a los servicios de presentación como HTML dinámico y Win32 para usuarios de intranets corporativas que esperan todos las bondades de los servicios de presentación de Windows, como a los servicios de presentación no interactivos como HMTL para soportar usuarios de Internet y extranets corporativas.

Para que un emisario soporte los servicios de presentación interactivos, éste debe soportar tanto validación a nivel propiedad (property-level), como a nivel objeto (object-level) En contraste para que un emisario pueda soportar los servicios de presentación no interactivos, solo necesita soportar validación a nivel objeto. La validación a nivel propiedad es el proceso de asegurar que una propiedad individual se encuentre dentro de un rango predefinido de valores. Por ejemplo, un emisario que soporte validación a nivel propiedad deberá asegurar que su propiedad CP contenga un Código Postal que éste definido por el Servicio Postal Mexicano. La validación a nivel propiedad le permite a los emisarios proporcionar retroalimentación inmediata a los servicios de presentación interactivos como aquellos creados utilizando HTML Dinámico o Win32. La validación a nivel objeto es el proceso de asegurar que todas las propiedades de un objeto se encuentren dentro de su predefinido rango de valores. Por ejemplo, un emisario que soporta validación a nivel objeto solo será válido si contiene un CP y un Estado válidos. Un Código Postal con un Estado no válido será por lo tanto un emisario no válido. La validación a nivel objeto permite a los emisarios proporcionar retroalimentación a los servicios de presentación no interactivos como aquellos creados utilizando HTML.

Confiabilidad

Confiabilidad se refiere a la habilidad de una aplicación de proporcionar resultados exactos. Una aplicación no es confiable si ésta regresa resultados inexactos. Sin embargo, el asegurar resultados exactos en un ambiente multiusuario es muy difícil. Por ejemplo, supongamos que una aplicación ésta diseñada para realizar transferencias monetarias de una cuenta a otra, abonando una cuenta y cargando a otra. Esto parece ser un a operación simple pero imagine los resultados si alguna parte del hardware o software del sistema fallara después de abonar a una cuenta y antes de cargar el dinero a la otra. O bien, imagine si dos clientes incrementan el balance de una cuenta en $50.00 como resultado de una compra reciente. Ambos clientes leen el balance actual $100.00 y entonces incrementan el balance actual en $50.00 (balance = balance + $50.00) para un balance final de $150.00 (el balance final debería ser $200.00) Para asegurar resultados exactos, deben llevar a cabo sus operaciones de negocios como una parte de una transacción manejada por *Transaction Server* (MTS) Las transacciones aseguran que las

transformaciones de estado sean Atómicas, Consistentes, Aisladas, y Durables (*Atomic, Consistent, Isolated, Durable ACID).*

Las operaciones Atómicas son aquellas que se completen enteramente o simplemente no se completan. Así, en el ejemplo anterior, tanto el cargo como el abono deben efectuarse para que la transformación de estado tenga éxito (tenga efecto), de otro modo la transformación de estado falla, y el sistema pasa a su último estado estable.

Las transformaciones consistentes, preservan la integridad interna de los recursos involucrados. Por ejemplo, el borrar registros de una tabla parent viola la integridad de la base de datos si es que existen registros relacionados.

Las transacciones aisladas parecen ocurrir de forma serial, una después de la otra, creando la ilusión de que ninguna otra transformación se está llevando a cabo en ese momento.

Durable se refiere a la habilidad de salvar los resultados de una transformación de estado, normalmente a disco, de manera tal que los resultados de una transformación puedan ser recuperados en el caso de una falla del sistema.

Debido a que las transacciones bloquean registros para asegurar un comportamiento ACID, deben ser consideradas como recursos críticos, lo que significa que a los clientes nunca se les debe permitir el acceso a éstas directamente. Imagine que pudiera pasar si un usuario iniciara una transacción y después abandonara la oficina para ir por una taza de café. Cualquier registro bloqueado resultado de la eminente transacción dejaría de estar disponible hasta que el usuario regrese a completar la transacción o que el tiempo de espera definido por el sistema se agote; así que para mantener la autonomía de una aplicación, las aplicaciones Windows DNA deben manejar las transacciones como recursos críticos.

Disponibilidad

La disponibilidad se refiere a la cantidad de tiempo que una aplicación es capaz de proporcionara para dar servicio a las peticiones del cliente, esto es importante porque una aplicación es útil solamente cuando sus servicios están disponibles a los clientes. La disponibilidad de una aplicación es dependiente de muchos factores que van mas allá del control del desarrollador, factores como la disponibilidad del hardware (discos duros, tarjetas NIC, controladores, etc.), la disponibilidad del software (RDBMS's, servidores Web, sistemas de espera, etc.), y la disponibilidad de la red. Para incrementar la disponibilidad del hardware y del software, las aplicaciones Windows DNA deben eliminar cualquier punto de falla potencial implementando sistemas redundantes. Las aplicaciones Windows DNA deben encontrarse sobre sistemas de hardware compuestos de arreglos de discos (RAID), múltiples NIC´s, múltiples controladores, etc., así como utilizar un esquema de cluster.

Escalabilidad

La escalabilidad es la utópica meta a alcanzar del rendimiento en el crecimiento lineal de los recursos adicionales, y es precisamente lo que permite a una aplicación soportar desde decenas de usuarios hasta decenas de miles de usuarios, con el solo hecho de adicionar o disminuir recursos de acuerdo a la necesidad de escalar una aplicación.

El rendimiento se refiere a la cantidad de trabajo, medido en transacciones (generado por usuarios o máquinas), que una aplicación puede llevar a cabo en un periodo de tiempo (normalmente segundos), y que es típicamente expresado en términos de transacciones por segundo (tps).

$$\text{Rendimiento} = \frac{\text{Transacciones}}{\text{Segundo}}$$

Por ejemplo, si una aplicación puede llevar a cabo en promedio una transacción en 20 milisegundos (ms), esa aplicación tiene un rendimiento de 50 transacciones por segundo, o 50 tps.

$$\frac{1\,\text{Transaccón X}}{20\,ms} = \frac{\text{Transacciones}}{1000\,ms}; \qquad X = 50\,\text{tps}$$

La escalabilidad es una medida del cambio total en el rendimiento resultante debido a un incremento en los recursos, específicamente los recursos necesarios requeridos para llevar a cabo las transacciones necesarias para obtener ese rendimiento. Por ejemplo, la aplicación A que tiene un rendimiento de 50 tps con X recursos y un rendimiento de 100 tps con 2X recursos tiene una escalabilidad mejor que la aplicación B que tiene un rendimiento de 50 tps con Y recursos, pero un rendimiento de 75 tps con 2Y recursos. Esto es porque el cambio total en el rendimiento de la aplicación A es 50 tps mientras el cambio total en el rendimiento de la aplicación B es solo 25 tps.

Debido a que la escalabilidad aumenta de forma proporcional al aumento del rendimiento (entre mayor sea el aumento del rendimiento por recurso, mayor será la escalabilidad), los desarrolladores de aplicaciones deben concentrar sus esfuerzos en incrementar el crecimiento del rendimiento, para así incrementar la escalabilidad. La llave para incrementar el crecimiento del rendimiento es reducir el tiempo total de la transacción, que es un producto de la cantidad de tiempo requerido para adquirir los recursos necesarios y la cantidad de tiempo que una transacción hace uso de esos recursos.

$$\text{Tiempo transacción} = \text{Tiempo de adquisición} + \text{Tiempo de uso del recurso}$$

Si a una transacción le toma 5 ms en adquirir la conexión a una base de datos y 10 ms en actualizar un registro de la base de datos, la transacción total tuvo una duración de 15 ms. Muchas cosas como la red, la velocidad de acceso a disco, el esquema de bloqueo de la base de datos (optimístico vs pesimístico), y la contención del recurso pueden afectar el tiempo de adquisición del recurso. De forma similar, muchas cosas como la red, la entrada del usuario, y el volumen de trabajo pueden afectar en el tiempo de uso del recurso. Incrementos en el tiempo de la adquisición y el uso del recurso incrementan el tiempo total de la transacción, lo que finalmente decrementa el rendimiento y la escalabilidad de la aplicación. Lo que significa que para incrementar la escalabilidad de una aplicación, los desarrolladores de aplicaciones Windows DNA deben concentrar en mantener los tiempos de adquisición y uso de los recursos tan cortos como sea posible. A continuación se mencionan algunos puntos a seguir para acortar los tiempos de adquisición y uso de los recursos.

- ✓ Evitar involucrar la interacción del usuario como parte de una transacción.
- ✓ Evitar la interacción de la red como parte de una transacción.
- ✓ Adquirir los recursos e inmediatamente reducirlos.
- ✓ Incrementar la disponibilidad de los recursos tanto como sea posible. Una alternativa es utilizar MTS para crear un "pool" de aquellos recursos que sean escasos o que sea costoso crearlos.
- ✓ Utilizar MTS para compartir los recursos entre los usuarios debido a que normalmente es más costoso crear un nuevo recurso que hacer uso de uno que ya existe.

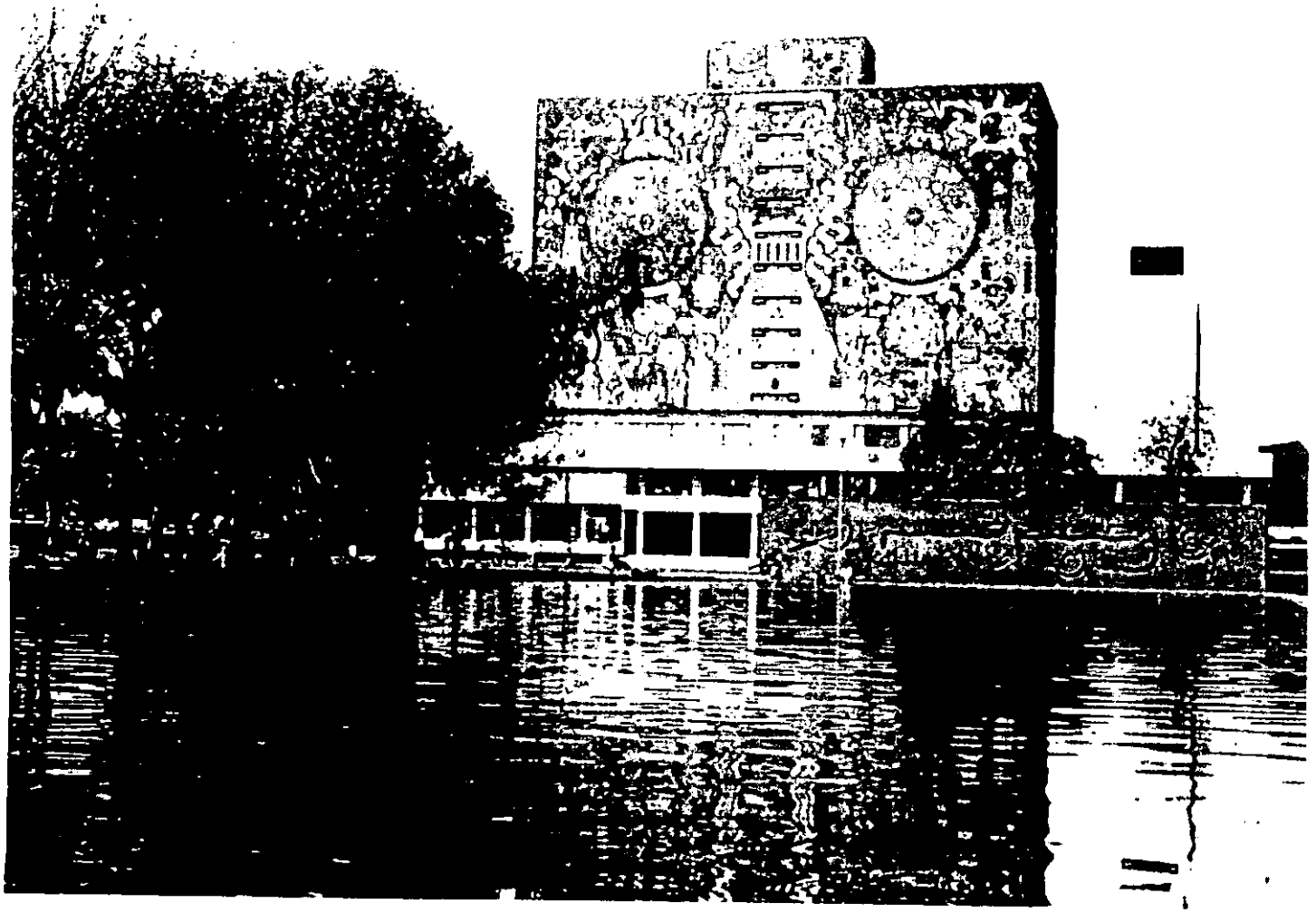
La implementación de múltiples servidores para el procesamiento del trabajo en una sola cola, es una forma eficiente de proporcionar un balanceo dinámico real de la carga. Debido a que cada servidor simplemente se mantiene procesando el trabajo de la cola tan pronto como le es posible, nunca se da situación de que un servidor se encuentre sobrecargado de trabajo mientras que otro ésta en espera de trabajo.

Interoperabilidad

La interoperabilidad se refiere a la habilidad de una aplicación para accesar otras aplicaciones, datos o recursos en otras plataformas. Muchos ambientes soportan diferentes y muy variados sistemas de hardware y software que deben trabajar en conjunto para funcionar exitosamente, por esto es importante que las aplicaciones Windows DNA sean interoperables. Para maximizar la interoperabilidad de una aplicación, las aplicaciones Windows DNA cuentan con:

Activex Data Objects (ADO) y OLE DB para el acceso universal a datos, Extensible Markup Language (XML) para compartir datos con otras aplicaciones, DCOM para el acceso a otras aplicaciones sobres sistemas UNIX y Multiple Virtual Storage (MVS); COM Transaction Integrator (COMTI) para ejecutar sistemas de control de información a clientes, (Costumer Information Control Systems CICS) o sistemas de manejo de información (Information Management Systems IMS).

Durante éste capítulo se abordo la arquitectura de Windows DNA para el desarrollo de aplicaciones, se definieron los principios sobre los cuales trabaja Windows DNA y se mencionaron brevemente las diferentes tecnologías que utiliza. Finalmente se profundizo en diseño de aplicaciones bajo el esquema de Windows DNA, para dar paso a definir los diferentes tipos de aplicaciones que existen, en el siguiente capítulo.



CAPÍTULO TRES

LA EVOLUCIÓN DE LAS APLICACIONES

Como se mencionó en el capítulo uno, un Sistema Nervioso Digital se construye a base de una combinación de hardware y software, y es precisamente la arquitectura de ese conjunto de software que corre sobre esas máquinas, a la cual se enfocará el contenido de este capítulo.

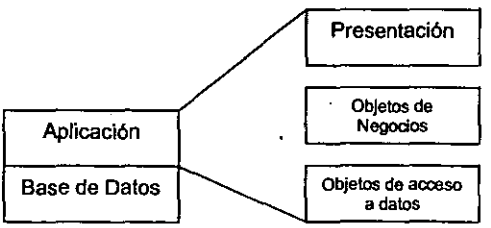
Cualquier aplicación, que permite la interacción con un usuario, por más simple que parezca, esta constituida de tres elementos esenciales: la presentación, la lógica de la aplicación y los datos.

La presentación se enfoca en la interacción con el usuario. La lógica de la aplicación lleva a cabo los cálculos y determina la conducta de la aplicación. Los elementos de datos administran la información que debe persistir a lo largo de las sesiones o que debe ser distribuida entre los usuarios.

Con respecto a la configuración y la implementación de aplicaciones, los desarrolladores tienen diversas opciones. Normalmente las aplicaciones de escritorio consisten en aplicaciones monolíticas que corren sobre una máquina. Las aplicaciones tradicionales de mainframe son también monolíticas, pero están distribuidas a lo largo de múltiples clientes. Algunas aplicaciones separan los elementos de presentación de la lógica de la aplicación y los datos. Aún más, otras aplicaciones conciben los tres elementos por separado. Estas capas no necesariamente corresponden a las localidades físicas sobre la red. Por ejemplo, las tres capas pueden implementarse sobre dos máquinas o ser implementadas sobre cinco, etcétera.

APLICACIONES DE UNA CAPA

Las aplicaciones de una sola capa son los modelos de aplicación más simples que existen. Este tipo de aplicaciones no son más que un programa, que corre en la máquina de un usuario. Aunque es un simple programa, es posible que haga algunos accesos a una base de datos, pero la base de datos reside en la misma máquina (o quizás se encuentre mapeada en un drive de red) Lo esencial en la definición de este tipo de aplicaciones es que, todo el trabajo se lleva a cabo en la misma máquina, es decir que tanto la interfaz de usuario, el proceso de negocios y de datos son efectuados en el mismo lugar.



Estructura de una aplicación de una capa

En este tipo de aplicaciones, todo es muy simple, ya que tanto la interfaz con el usuario, como la lógica de aplicación y los datos se encuentran interactuando en una misma máquina.

APLICACIONES DE DOS CAPAS CLIENTE/SERVIDOR

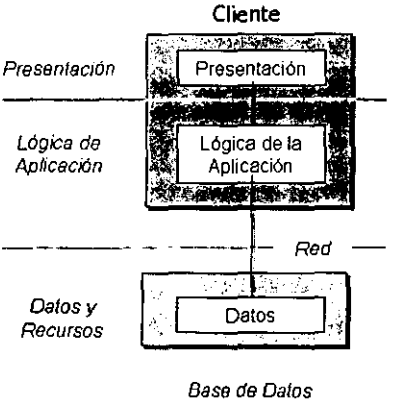
Las aplicaciones de dos capas Cliente/Servidor, agrupan la presentación y los componentes de la lógica de la aplicación en la máquina cliente y accesan a los datos usando una conexión de red.

La ventaja de tales configuraciones es que los datos están centralizados. Esta centralización beneficia a la organización en los siguientes puntos:

- Los datos son compartidos
- Proporciona consistencia en los datos accesados
- Reduce la duplicación y el mantenimiento
- Simplificación en la generación de reportes
- Facilidad en la actualización de hardware

Las aplicaciones de dos capas trabajan bien en aplicaciones de escala departamental con un número modesto de usuarios (menos de 100), una sola base de datos, y una red rápida y segura.

Por ejemplo, una aplicación de dos capas es un beneficio obvio para un grupo de trabajo cuya función es introducir órdenes de producto. Otro ejemplo es un departamento de diseño que comparte un conjunto de diseños entre su grupo de ingenieros.



Estructura de una aplicación de dos capas

Limitaciones de las aplicaciones de dos capas.

Las arquitecturas clásicas de dos capas han brindado eficiencia a los negocios, pero existe también un número de limitantes:

LIMITANTE	EXPLICACIÓN
Aplicaciones cliente monolíticas	Las aplicaciones de dos capas tienden a tener componentes del lado del cliente, lo cual impide mejoras increméntales (actualizaciones y corrección de "bugs") para la aplicación.
Dificultad para escalar	El escalamiento de las aplicaciones es pobre debido a que el número de conexiones disponibles a la base de datos esta limitado. Las peticiones de conexión más allá de este límite son simplemente rechazadas.
Dificultad para el mantenimiento	Es difícil mantener la lógica de la aplicación del lado del cliente debido a que debe de ser implementada en cada cliente. Cualquier cambio en la lógica debe ser redistribuido a todos los clientes.
Confidencialidad comprometida	La lógica de la aplicación del lado del cliente expone potencialmente las reglas de negocios a los usuarios.
Dificultad para la reutilización	Es difícil reutilizar la lógica de una aplicación de dos capas, debido a que las aplicaciones esta vinculadas a bases de datos y formatos de tablas específicos.
Estricta vinculación a las fuentes de datos	El cliente esta normalmente configurado para una base de datos en particular, así que mover datos a una base de datos diferente es más difícil. (ODBC datasources resuelven el problema en alguna forma)
Rendimiento pobre de la red	La red corre ineficientemente debido a la cantidad de datos que son transferidos a lo largo de esta. La mayoría de los procesamientos de bases de datos no están localizados.

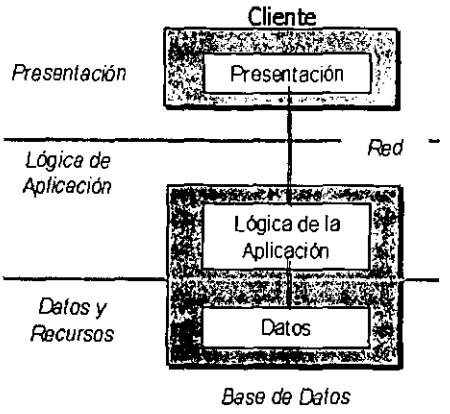
El papel de los stored procedures en las aplicaciones de dos capas

Las arquitecturas de dos capas demuestran los beneficios de la centralización de datos. Las limitaciones del modelo estándar de dos capas han motivado la investigación y desarrollo de otras alternativas. Resumiendo, estas limitaciones son:

- ✓ Clientes pesados que hacen el mantenimiento y reutilización problemáticos
- ✓ Las bases de datos que escalan pobremente más allá de cierto número de usuarios
- ✓ Demasiados datos enviados sobre la red para ser procesados en el cliente

Algunas de estas limitaciones tienen solución haciendo uso de *stored procedures*. Los *stored procedures* son funciones precompiladas que se administran y corren dentro de una base de datos, éstos ofrecen los siguientes beneficios a las aplicaciones:

BENEFICIO	EXPLICACIÓN
El procesamiento de datos ocurre en la fuente de datos	Debido a que están almacenados en la base de datos, los *stored procedures* pueden procesar los datos en la fuente de forma más rápida que transfiriendo los datos a lo largo de la red. Por ejemplo, en lugar de calcular un valor promedio transfiriendo el dato a través de la red y generando el valor en el cliente, un procedimiento almacenado puede calcularlo de forma local.
Mejora en la seguridad	El mejoramiento de la seguridad es posible ya que se restringe el acceso a los *stored procedures* y se denega el acceso a la estructura de los datos. Además, esto encapsula los datos al grado que la estructura puede cambiar sin romper el código del lado del cliente.
La lógica de procesamiento es distribuida a través de múltiples aplicaciones	La reutilización de la aplicación es mejorada permitiendo que muchos componentes de presentación puedan llamar al mismo *stored procedure*.
Escalabilidad mejorada	Los procedimientos almacenados mejoran la escalabilidad de la base de datos permitiendo al código especializado y propio de la base de datos optimizar el proceso de datos. Estos procedimientos almacenados son compilados para correr rápida y eficientemente.



Estructura de una aplicación de dos capas utilizando procedimientos almacenados

Limitaciones de los *STORED PROCEDURES*

A pesar de las ventajas que los *stored procedures* ofrecen, existe un número significativo de limitaciones:

- ✓ SQL, el lenguaje en que los *stored procedures* son programados, no es tan rico como la mayoría de los lenguajes de programación como Visual Basic, Visual C++, Visual J++ y Visual Fox Pro.
- ✓ Los procedimientos almacenados que consuman gran cantidad de tiempo de proceso pueden obstruir el acceso a la base de datos.
- ✓ Los *stored procedures* deben correr sobre la misma máquina que la de los datos, limitando el escalamiento de una aplicación.
- ✓ Los *stored procedures* se diseñan para un RDBMS particular y no pueden ser fácilmente movidos a otro diferente.

LAS APLICACIONES CLIENTE/SERVIDOR DE TRES CAPAS.

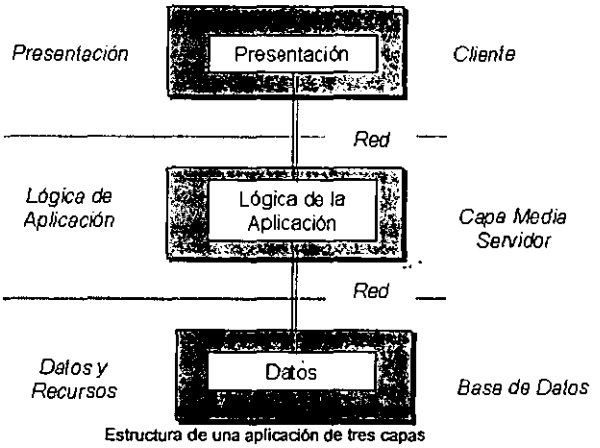
Con el tiempo, se ha hecho aparente que la arquitectura de dos capas Cliente/Servidor no es lo suficientemente poderosa o flexible para manejar grandes aplicaciones. Debido a que cada cliente mantiene una comunicación constante con el servidor central de datos, el tráfico de la red se incrementa considerablemente. Más aún, el servidor central de bases de datos se puede convertir en un cuello de botella en tanto se incrementa el número de usuarios que tratan de ingresar simultáneamente al mismo recurso.

El esquema Cliente/Servidor de tres capas ayuda a solucionar este problema, anteponiendo físicamente otro servidor entre el cliente y la base de datos. Este *servidor central de aplicaciones* puede manejar eficientemente el tráfico sobre la red y la carga sobre el servidor de bases de datos. En muchas ocasiones se tiene una conexión de red

mucho más rápida entre el servidor de aplicaciones y el servidor de bases de datos, y de esta forma se incrementa más aún la eficiencia en la aplicación.

En las arquitecturas de tres capas, la presentación, la lógica de la aplicación y los elementos de datos están conceptualmente separados. Los componentes de presentación administran la interacción del usuario y solicitan servicios de aplicación mediante el llamado a componentes de capas intermedias. Los componentes de aplicación llevan a cabo la lógica de negocios y realizan peticiones a la base de datos.

El diseño de la aplicación se hace más flexible debido a que los clientes pueden hacer tantas llamadas a los componentes como sean necesarias para completar una petición, y los componentes pueden llamar a otros componentes para mejorar la reutilización del código.



Estructura de una aplicación de tres capas

Ventajas de las aplicaciones de tres capas

VENTAJA	EXPLICACIÓN
Soporte multilenguaje	Los componentes de aplicación pueden ser desarrollados usando lenguajes de propósito general.
Componentes centralizados	Los componentes pueden estar centralizados para un fácil desarrollo, mantenimiento y distribución.
Balanceo de la carga	Los componentes de aplicación pueden estar ubicados en múltiples servidores, permitiendo una mejor escalabilidad
Acceso más eficiente a los datos	Los problemas de limitación de conexiones a las bases de datos son minimizados debido a que la base de datos sólo ve al componente, y no a todos los clientes. También las conexiones a la base de datos y los drivers no son requeridos del lado del cliente. Las conexiones a la base de datos en las aplicaciones de

	dos capas son *early adquired* y se mantienen en todo el tiempo de ejecución de la aplicación, mientras que en las aplicaciones de tres capas son *late adquired* y se liberan inmediatamente.
Mejoramiento de la seguridad	Los componentes de capas intermedias pueden estar asegurados centralmente usando una infraestructura común. Los accesos pueden ser otorgados o denegados componente a componente, simplificando la administración
Acceso simplificado a recurso externos	El acceso a recursos externos, tales como aplicaciones mainframe y otras bases de datos, esta simplificado; un servidor *gateway* se transforma en otro componente que puede ser usado por la aplicación.

APLICACIONES DISTRIBUIDAS

Antes de definir que es una aplicación distribuida, vale la pena definir más a fondo que es una aplicación.

Una aplicación es un programa de computadora que es usado para resolver un problema en particular o un conjunto de problemas relacionados. Las aplicaciones más básicas corren en un espacio de proceso propio y a veces se valen de utilerías o funciones auxiliares (*DLL´s Dynamic Link Libraries*) para llevar a cabo su propósito.

Los servidores son aplicaciones especializadas que proveen servicios a otras aplicaciones. En tal contexto, la aplicación que solicita un servicio se conoce como Cliente. Clientes y servidores normalmente corren en máquinas separadas conectados por una red. Una aplicación cliente usando los servicios de una aplicación servidor es lo que normalmente se conoce como aplicación Cliente/Servidor.

Frecuentemente, los servidores se comunican simultáneamente con múltiples clientes. Para el cliente, el servidor parece estar interactuando exclusivamente con él. En realidad, los servidores son recursos distribuidos que son usados por muchos clientes.

Para satisfacer la petición de un cliente, un servidor debe convertirse en cliente de uno o más servidores, creando una cadena de procesamiento que va del cliente original a muchos servidores. Esto es lo que conocemos como una aplicación distribuida. Una aplicación distribuida esta compuesta de múltiples aplicaciones especializadas trabajando juntas para alcanzar una meta útil.

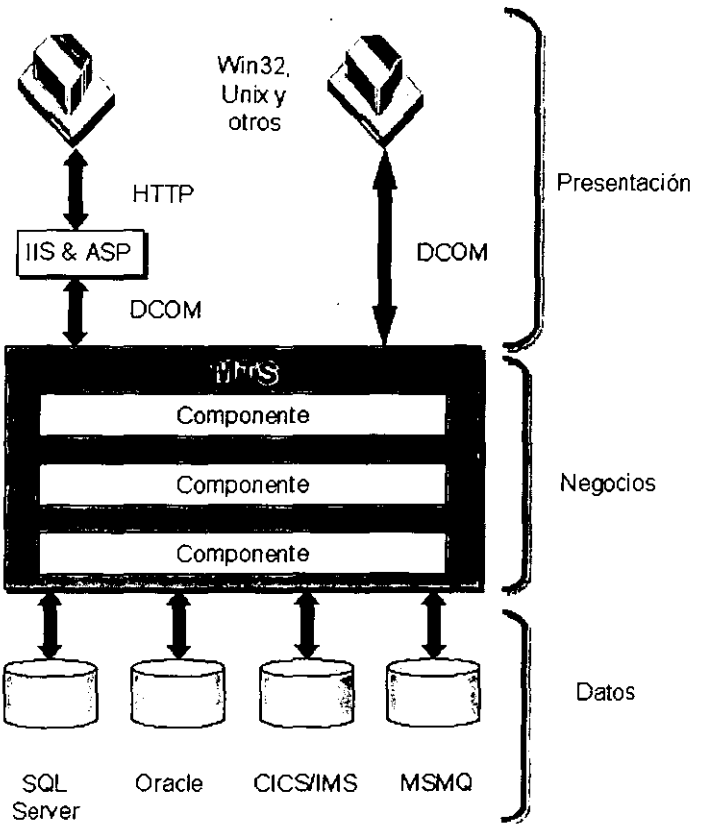
El termino "*n-tier application*" o aplicaciones de n-capas es otra forma de referirse a las aplicaciones distribuidas. Una aplicación n-capas representa un caso especial de aplicación de tres capas, donde una o más de estas capas pueden están separadas en capas adicionales, lo cual suministra otra capa de abstracción.

Una aplicación distribuida lógicamente, divide los componentes de un sistema Cliente/Servidor en tres capas: datos, lógica de negocios y presentación. Estas capas no necesariamente corresponden a ubicaciones físicas en la red.

La razón para utilizar aplicaciones distribuidas es muy simple, al particionar una aplicación compleja en secciones de presentación, lógica de aplicación y datos, el resultado será un aumento en la eficiencia, escalabilidad y un posible aumento en la confiabilidad del sistema.

Al diseñar aplicaciones de esta forma, se proporciona mayor flexibilidad cuando se implementan procesos y datos en la red, además de un incremento en el rendimiento y en la facilidad de uso.

La lógica de presentación, la lógica de negocios y la lógica de bases de datos en una aplicación distribuida pueden correr sobre diferentes computadoras.



Estructura de una aplicación distribuida y algunas de las tecnologías involucradas en su implementación.

Para extender este concepto a un nivel superior, una aplicación distribuida a nivel empresarial es una aplicación robusta distribuida que es flexible a las necesidades

cambiantes de los negocios y escalable a la demanda de crecimiento del usuario. Puede ser implementada sobre una variedad de plataformas a lo largo de redes corporativas, intranets o Internet. Debe ser amigable al usuario y debe satisfacer estrictamente los requerimientos de seguridad, administración y mantenimiento. En concreto, es un sistema altamente complejo.

Beneficios de las aplicaciones distribuidas

Como cualquier otra aplicación moderna, una aplicación distribuida debe proporcionar una interfaz intuitiva para el usuario, así como satisfacer las necesidades de los usuarios. Mas allá de estas cualidades comunes, también puede estar caracterizada por tres requerimientos adicionales.

Una aplicación distribuida es:

✓ Escalable.
En tanto el número de usuarios o carga de trabajo se incremente, el desempeño de la aplicación no se debe degradar significantemente.

✓ Confiable
Las aplicaciones confiables no obstaculizan el trabajo de los usuarios, debido a fallas del software o hardware.

✓ Eficiente
Las aplicaciones eficientes hacen su trabajo rápidamente y son útiles al usuario para alcanzar su meta.

Una aplicación distribuida a nivel empresarial, o una aplicación empresarial, es una aplicación distribuida de gran escala. Las aplicaciones empresariales tienen características adicionales más allá de una aplicación distribuida. Una aplicación empresarial es:

✓ Extensiva
En la empresa es común encontrarse con una aplicación multi-usuario, multi-máquina que puede manipular enormes cantidades de datos, utilizar procesamiento en paralelo, recursos de red distribuidos y lógica compleja. Esta aplicación puede ser implementada sobre múltiples plataformas e interoperar con muchas otras aplicaciones. Las aplicaciones empresariales son de larga vida.

✓ Orientada a los negocios
El propósito es satisfacer requerimientos de negocio específicos. Involucra políticas de negocios, procesos, reglas y entidades, es desarrollada en una organización e implementada de forma sensible a las necesidades de los negocios.

✓ Misión crítica
Una aplicación empresarial debe ser lo suficientemente robusta para mantenerse en operación continua. Debe ser extremadamente flexible para su escalabilidad e implementación, y permitir un mantenimiento, monitoreo y administración efectivos.

Beneficios tácticos

Los beneficios tácticos (a corto plazo) de las aplicaciones distribuidas son:

- ✓ La estrategia de divide y vencerás puede ser usada en el desarrollo de la aplicación
- ✓ Posibilidad para centralizar el manejo de las reglas del negocio
- ✓ Uso eficiente del pool de conexión
- ✓ Distribución flexible de datos (los clientes no saben en donde están los datos)
- ✓ Máxima utilización del servidor (muchos clientes pueden estar usando un servidor)
- ✓ Mejor modelo de seguridad (roles)
- ✓ Facilidad al interfasar protocolos (entre el cliente y la base de datos)
- ✓ Los clientes se pueden personalizar (Navegadores de Internet y clientes de escritorio accesando los mismos componentes de capas intermedias)

Beneficios estratégicos:

Los beneficios estratégicos (a largo plazo) de las aplicaciones distribuidas son:

- ✓ Incremento en la eficiencia del trabajador (mejores resultados)
- ✓ Mejores trabajadores a través de sus facultades
- ✓ Nuevo conocimiento del negocio (vía data mining por ejemplo)

TERMINOLOGÍA Y CONCEPTOS

En esta parte del capítulo se define que tan importantes son los atributos propios de los componentes en el desarrollo de una aplicación distribuida.

Componentes

Los componentes son unidades de software que proveen funcionalidades específicas para una aplicación. Hay muchas razones para usar componentes en una aplicación:

- ✓ Reusabilidad

  Una vez creado un componente, otros desarrolladores pueden usarlo. Esto habilita un fácil acceso a las características de los componentes en otras aplicaciones sin requerir que los desarrolladores escriban código extensivo.

- ✓ Reducción de la complejidad

  Se puede crear un componente que oculte la complejidad de la programación para otros programadores, ya que éstos sólo necesitan conocer la información que recibe el componente y el resultado que arroja.

- ✓ Actualización más sencilla

  Los componentes facilitan la revisión y actualización de las aplicaciones. Por ejemplo, se puede crear un componente que encapsule las reglas de negocio. Si las reglas de negocio cambian, se actualiza sólo el componente, y no todas las aplicaciones que utilizan el componente.

- ✓ Un desarrollo más sencillo

  Separando el código en componentes da la oportunidad de desarrollar y probar piezas encapsuladas de la aplicación en forma independiente. También permite que en un proyecto las tareas se distribuyan entre los miembros del equipo de desarrollo.

- ✓ Neutralidad del lenguaje

  Los componentes pueden interactuar unos con otros, independientemente del lenguaje de implementación. Esto permite al equipo de desarrollo usar de forma más eficiente el lenguaje para implementar la funcionalidad de los componentes.

Las aplicaciones distribuidas sobre la plataforma de Windows y otras plataformas son posibles gracias a COM (Component Object Model) y DCOM (Distributed Component Object Model), estas tecnologías permiten a los componentes comunicarse entre ellos.

COM proporciona un mecanismo que permite a los componentes de código binario, derivados de cualquier combinación de componentes preexistentes y componentes de diferentes distribuidores de software, conectarse y comunicarse unos con otros de una forma bien definida. Usando COM, se pueden construir diferentes componentes que trabajen juntos en una sola aplicación.

DCOM permite a los componentes COM comunicarse directamente unos con otros a lo largo de la red.

Transacciones

El trabajo realizado en una aplicación distribuida multiusuario puede involucrar cambios a tablas de una base de datos a la par que a entidades ajenas a la base de datos. Una transacción involucra un conjunto de tareas que deben llevarse a cabo en su totalidad para proporcionar una consistencia en el trabajo realizado. Debido a que la tarea encomendada a un objeto puede fallar, la aplicación debe estar habilitada para manejar situaciones en donde algunos objetos fallen, y otros no. El sistema no puede ser dejado en un estado inconsistente donde parte del trabajo esta hecho, pero otra parte no.

Las transacciones proveen un modelo simple de manejo de trabajo, este modelo acepta dos situaciones: se cumple la transacción en su totalidad o no se cumple nada. Ya sea que todos los objetos tengan éxito y la transacción sea aceptada, o bien que uno o más de los objetos fallen y la transacción sea desechada

Una transacción provee un servicio tolerante a fallas (fault-tolerance) para un conjunto de recursos computacionales y es elemento esencial para una aplicación distribuida de nivel empresarial.

Las transacciones permiten a los participantes, recursos y procesos, dar su voto en el resultado de una operación transaccional. Un participante, tal como un manejador de bases de datos, puede votar por éxito o fracaso. Un voto de fracaso de parte de cualquier participante invalida la transacción, llevando al sistema al estado original en el que se encontraba en el momento que dió comienzo la transacción. Si todos los participantes votan por éxito, la transacción tendrá éxito y el sistema se llevará a un nuevo estado.

Las transacciones, así como los monitores de procesamiento de transacciones (TP monitors), reemplazan a los métodos hechos a la medida para proporcionar un ambiente confiable con una infraestructura bien definida que pueden ser usados sobre múltiples máquinas y procesos.

En la misma manera que los componentes proporcionan una forma de manejar el desarrollo estático de aplicaciones, las transacciones ayudan a los desarrolladores a manejar la complejidad del tiempo de ejecución de las aplicaciones. Las transacciones encapsulan los detalles de las operaciones en tiempo de ejecución y proporcionan un resultado de aceptación o rechazo que puede ser aprovechado para simplificar el proceso de una aplicación distribuida.

Una definición más amplia sobre estas tecnologías lo encontramos en el capítulo de ***La plataforma y los servicios.***

Por último es conveniente resaltar que a lo largo de este capítulo, y en particular en su segunda mitad, se hace énfasis en la arquitectura de las aplicaciones distribuidas, debido a que este esquema es el que se considero más apropiado para el desarrollo de un Sistema Nervioso Digital, aunque se podrá notar a lo largo de la tesis que si bien es el más adecuado no significa necesariamente que sea el más sencillo en su desarrollo.

CAPÍTULO CUATRO

LA PLATAFORMA Y LOS SERVICIOS

Hasta este momento hemos ya definido lo que es un Sistema Nervioso Digital, así también hemos visto que la arquitectura de Windows DNA permite el desarrollo eficiente de aplicaciones distribuidas. El siguiente paso en el desarrollo de un DNS depende de la plataforma, las herramientas y los servicios que permitirán el funcionamiento y la interconexión de los diferentes subsistemas que forman parte de un DNS. En este capítulo se explican los servicios más importantes que la arquitectura de Windows DNA define. Vale la pena mencionar que los servicios aquí presentados no son todos los servicios sobre los que trabaja una aplicación Windows DNA, debido a que el número de servicios utilizados es muy extenso y el hablar que cada uno requiere de una explicación detallada para poder entender su interrelación con otros servicios. Sólo se abundará en la explicación de los servicios de mayor importancia o esenciales para el funcionamiento de la aplicación.

EL SERVIDOR WEB (INTERNET INFORMATION SERVER)

Entre sus diferentes papeles, la Web ha emergido como un vehículo ideal para proporcionar a los usuarios aplicaciones de negocios y de comercio electrónico. Los navegadores representan una forma fácil de proporcionar interfaces de usuario dinámicas y poderosas con costos de mantenimiento casi nulos.

La presentación y la interacción con el usuario conforman sólo una parte de la aplicación total. Necesita existir algo extra que tome el código HTML e invoque una aplicación que procese apropiadamente las peticiones sobre las bases de datos y los recursos. En el caso de la Web ese algo es normalmente un servidor Web. En un principio los servidores Web proporcionaban acceso a páginas estáticas de HTML, la mayoría de los servidores Web ahora cuentan con mecanismos para aceptar peticiones del usuario, invocar una aplicación o un script, y regresar una respuesta.

El proceso de manejar las peticiones de aplicación para los servidores Web es más difícil de lo que parece. La mayoría de los servidores Web ofrecen nada o muy poco de infraestructura para el manejo de aplicaciones por sí mismos. Pueden invocar una aplicación o un script, pero depende del desarrollador el proporcionar los servicios básicos tales como el acceso a bases de datos.

La mayoría de las tecnologías de desarrollo de aplicaciones ofrecen una buena infraestructura de servicios aunque no bien integrada con los servidores Web. Por ejemplo, un problema común es mantener información sobre la parte del usuario entre una y otra petición del navegador en el mundo desconectado de la Web.

Soluciones convencionales a estos inconvenientes han recaído en mecanismos tales como los scripts, la *Common GateWay Interface (CGI)* y los "*cookies*", que funcionan bien para aplicaciones básicas, pero que no son escalables ni lo suficientemente robustos para aplicaciones de línea de negocios o de comercio electrónico.

El *Internet Information Server (IIS)* es un servidor de archivos y aplicaciones que opera bajo el sistema operativo Windows NT. IIS soporta protocolos de información estándar y

es altamente extensible a través del uso de *Internet Application Programming Interface (ISAPI)* y de *Common Gateway Interface (CGI)*. IIS proporciona una solución de servicios de sitios en Internet, intranet y extranet.

IIS es el componente base para el desarrollo de una aplicación de Internet o intranet. Además se encuentra altamente integrado con Windows NT Server por lo que toma ventaja de la seguridad integrada en este y del *Windows NT File System (NTFS)*.

Con IIS, se pueden implementar aplicaciones escalables para alojar las ultimas tecnologías de generación del contenido Web. IIS soporta las herramientas de programación como Visual Basic, VBScript, Jscript, así como componentes Java. También soporta aplicaciones CGI además de extensiones y filtros ISAPI.

EXTENSIONES DEL SERVIDOR WEB

IIS aumenta las capacidades de Windows NT, principalmente al dar la oportunidad de publicar en una amplia variedad de formatos. Con IIS se pueden correr aplicaciones que no son nativas de un servidor Web, tales como el acceso a bases de datos, e incrustar scripts en páginas HTML. Estas aplicaciones incluyen: *Active Server Pages (ASP)*, *Common Gateway Interface (CGI)*, *Internet Service Application Interface (ISAPI)*, y *Open Data Base Connectivity (ODBC)*.

ASP (ACTIVE SERVER PAGES)

ASP permite incrustar scripts en documentos HTML comunes y corrientes. Estos scripts pueden ser usados para ejecutar la lógica de la aplicación e invocar componentes de software que lleven a cabo tareas especializadas, tales como consultas a bases de datos, entrada/salida de archivo (I/O), reglas de negocio y flujo de trabajo. ASP combina la simplicidad del *Internet Database Connector (IDC)* con la flexibilidad de ISAPI, y debido a que soporta *Java Virtual Machine (JVM)*, se pueden crear ASP's en una variedad de lenguajes como Java y Jscript.

CGI (COMMON GATEWAY INTERFACE)

CGI es la extensión del servidor Web usada mas comúnmente. CGI permite correr aplicaciones no nativas al servidor Web. Por ejemplo, una vez que un usuario complete una forma HTML, se puede usar CGI para pasar la información proporcionada por el usuario a una aplicación remota para su procesamiento, y posteriormente obtener los resultados de la aplicación remota y regresarlos al usuario por medio de una página HTML.

Muchas de las aplicaciones CGI están escritas en lenguajes de scripting tales como *Practical Extraction and Report Language (PERL)*, que es un lenguaje interpretado muy parecido a Basic. Debido a su portabilidad, estos lenguajes son una manera popular de extender las capacidades del servidor Web, de tal forma que se puede copiar cualquier script en PERL que se encuentre corriendo en un servidor Web sobre UNIX, y correrlo directamente sobre el IIS. Por otro lado las aplicaciones binarias necesitarían ser recompiladas.

A pesar de su flexibilidad y portabilidad, las aplicaciones PERL no son la solución óptima para los sitios Web que tienen un tráfico pesado debido a que CGI debe iniciar un nuevo proceso por cada petición CGI. Después de que CGI da servicio a la petición, elimina ese proceso y toda información asociada con éste. Por ejemplo, si se tiene un script que proporciona al usuario el número de veces que ha sido accesada la página actual, y se tiene un trafico pesado, este proceso puede ocasionar problemas en el rendimiento debido a que CGI no pone la información en "cache".

ISAPI (INTERNET SERVER APPLICATION PROGRAMMING INTERFACE)

La Internet *Server Application Programming Interface (ISAPI)*, fue creada para suplir a CGI. ISAPI es un conjunto de rutinas extensibles de propósito general para crear llamadas a aplicaciones externas y manipular el flujo de datos entre el navegador y el servidor.

ISAPI es una especificación abierta soportada por servidores Web de terceros sobre Windows NT y otros sistemas operativos. Con la combinación de ISAPI, IIS y Windows NT, se puede crear una plataforma HTTP de alto rendimiento a bajo costo.

Las aplicaciones ISAPI son usadas para solucionar los problemas de rendimiento ocasionados por CGI. Además, los filtros ISAPI son usados para pre-procesar y post-procesar mensajes que entran y salen del servidor Web.

ODBC (OPEN DATA BASE CONNECTIVITY)

IDC es otra opción de extensibilidad incluida en el IIS. IDC es una aplicación ISAPI que permite conectar una página Web a cualquier base de datos que soporte ODBC.
Este conector ODBC permite tomar ventaja de las soluciones de bases de datos existentes incluyendo:

- ✓ Un machote en HTML que permite mostrar los resultados de las consultas a la base de datos.
- ✓ Desarrollo interactivo de aplicaciones con fuentes de datos ODBC
- ✓ No requiere de programación: se crea una consulta y se proporciona el machote de salida para conectarlo a la fuente de datos.
- ✓ Alto rendimiento debido a que es una aplicación ISAPI.

El hecho de que IDC utiliza un lenguaje de scripting simple para crear conexiones a bases de datos, lo hace una extensión del servidor particularmente popular.

EL MANEJO DE TRANSACCIONES (MICROSOFT TRANSACTION SERVER)

Una vez expuesto lo referente al servidor de Web, que es el servicio que nos permitirá postear las páginas HTML o ASP, toca el turno al servicio que permite el manejo transaccional de las operaciones que son llevadas a cabo como parte del funcionamiento normal de un DNS.

Debido a que el *Transaction Server (MTS)* soporta el manejo de aplicaciones basadas en componentes, permite la separación de la lógica de negocios que puede ser implementada en componentes ActiveX. Los cambios en la lógica del negocio pueden llevarse a cabo con gran facilidad y sin afectar a otras partes de la aplicación. Además, los componentes ActiveX permiten la reutilización de código, de tal forma que al crear

aplicaciones se puede hacer uso de componentes previamente creados o bien de terceros.

Por otra parte, MTS ofrece numerosas características que son transparentes para el programador y que eliminan la compleja programación de bajo nivel, permitiendo al desarrollador enfocarse esencialmente en la lógica de negocio de la aplicación. Como resultado el programador realiza menos trabajo de programación y por lo tanto reduce el tiempo de desarrollo.

MTS esta basado completamente en COM, de esta forma utiliza una tecnología robusta y bien conocida, algo de gran importancia cuando se desarrollan aplicaciones a nivel empresarial. Debido a que esta basado en COM, no necesita un ambiente de tiempo de ejecución adicional en las máquinas clientes, por esto podemos instanciar e invocar métodos de componentes en MTS, de la misma forma que lo hacemos para cualquier otro componente.

TAREAS COMPLEJAS QUE MANEJA EL MTS

- ✓ Transacciones automáticas
- ✓ Balanceo de cargas estático
- ✓ Unión de recursos y *multi-threading*
- ✓ Tolerancia a fallas
- ✓ Concurrencia multiusuario
- ✓ Seguridad basada en roles

MTS ES EN REALIDAD DOS SERVICIOS EN UNO:

- Es un Corredor de Peticiones de Objetos *(ORB Object Request Broker)*. Maneja instancias de componentes y servicios para estos objetos en un ambiente de ejecución configurable
- Es un procesador de transacciones. Trabaja en conjunto con el servicio de *Distributed Transaction Coordinator (MS DTC)* para coordinar transacciones a través de diversas fuentes de datos.

Es importante mencionar que existen dos tipos componentes, igual de importantes, que se ejecutan en el MTS, y éstos son los componentes transaccionales y los no transaccionales. Ambos tipos de componentes se benefician de los servicios que provee el MTS en tiempo de ejecución.

A pesar de su nombre, MTS, se considera que su capacidad para el proceso de transacciones es la menos importante de sus características. Sin embargo, las transacciones juegan un papel de vital importancia en el desarrollo de aplicaciones de *n* - capas que hacen manejo de datos distribuidos, usuarios distribuidos y componentes distribuidos.

MTS simplifica el desarrollo e implementación de aplicaciones. MTS es ideal para el desarrollo de aplicaciones de comercio electrónico y de línea de negocios con interfaz basada en el Web, utilizando herramientas de desarrollo como Visual Basic para el desarrollo en tres capas, y la construcción de aplicaciones de calidad con la tecnología *Distributed COM (DCOM)*

Podemos dividir los servicios de MTS en cuatro categorías principales:

Primero, existe un Corredor de Peticiones de Objetos *(Object Request Broker ORB)*. Cuando una llamada llega al servidor requiriendo un objeto, el ORB se encarga de esta llamada, verifica la disponibilidad, y por último despacha el objeto solicitado.

Después, un Monitor de Procesamiento de Transacciones *(Transaction Processing Monitor -TP Monitor)* que en términos simples es un ambiente que se inserta a sí mismo entre los recursos del servidor y de los clientes de tal forma que pueda manejar las transacciones, administrar los recursos, y proporcionar balanceo de cargas y tolerancia a fallas. El TP Monitor típico no reconoce a otros objetos, solamente sabe como manejar las peticiones en la mejor forma posible.

MTS combina sus características de ORB y TP Monitor soportándose en el *Modelo Distribuido de Objetos Componentes (DCOM - Distributed Component Object Model)* como elemento principal. MTS usa el Coordinador de Transacciones Distribuidas *(DTC - Distributed Transaction Coordinator)* como TP Monitor.

En un proceso típico, las peticiones en busca de objetos vienen por medio de DCOM, MTS procesa las peticiones, acomoda eficientemente los recursos, e inicializa al DTC. El DTC regresa un objeto al cliente. El cliente usa el objeto para llevar a cabo sus tareas. MTS reside entre el objeto y el cliente y monitorea todas las acciones de este último. Al hacer esto, actúa como un TP Monitor y como resultado, puede llevar a cabo actividades del TP Monitor. Por ejemplo, puede calendarizar peticiones y crear un "pool" de recursos en su tiempo libre. Cuando el cliente ha terminado con el objeto, el cliente lo libera lo que ocasiona que el MTS complete la transacción y libere o cree un "pool" de recursos.

MTS es la mejor tecnología para el desarrollo de aplicaciones distribuidas sobre Windows NT por las siguientes razones:

- ✓ MTS es la mejor forma de implementar aplicaciones basadas en COM sobre Windows NT ya que los desarrolladores pueden utilizar herramientas populares de desarrollo como Visual Basic. La arquitectura de MTS basada en componentes, simplifica el uso y reutilización de objetos, además de permitir a los administradores una administración gráfica sencilla.

- ✓ MTS ofrece una funcionalidad extensa de componentes, tal como el manejo automático de transacciones, un esquema de seguridad simple pero poderoso al mismo tiempo basado en roles, acceso a las bases de datos más comerciales, encolamiento de mensajes y aplicaciones mainframe, además de mejoras en el rendimiento como el "pool" de conexiones a bases de datos.

- ✓ MTS se encuentra totalmente integrado con otras aplicaciones tales como IIS y ASP para facilitar el desarrollo de aplicaciones Internet e intranet, los servicios de agrupamiento de Windows NT para la tolerancia a fallas, y el esquema de seguridad de Windows NT para el control simplificado de los recursos.

Tabla 4.1 Características del MTS

CARACTERÍSTICA	DESCRIPCIÓN Y BENEFICIOS
FACILIDAD DE USO	
Simplifica el desarrollo de aplicaciones de n capas	MTS separa completamente la infraestructura de presentación y la lógica de la aplicación. Esto permite a los desarrolladores construir aplicaciones como si fueran una colección de componentes individuales e implementarlos en la capa apropiada.
Soporte para componentes COM	MTS soporta cualquier herramienta que pueda producir componentes bajo el estándar COM.
Manejo simple de interfaces de programación	Es necesario aprender solo dos API´s para poder implementar los componentes en el ambiente de ejecución del MTS
Soporte para paquetes de componentes	Es posible juntar los componentes relacionados en paquetes. Los paquetes proveen una forma fácil y segura de manejar e implementar los componentes como grupo.
MTS Explorer	El MTS Explorer es una interfaz gráfica que facilita la administración de los paquetes y de los componentes.
Utilería de instalación automatizada del cliente	Esta utilería simplifica y automatiza el proceso de configuración de las máquinas cliente que requiere del acceso a los componentes que se ejecutan en el servidor.
FUNCIONALIDAD	
Manejo automático de transacciones	MTS proporciona un manejo automático de las transacciones. Los desarrolladores o los administradores solo tienen que definir que nivel de protección transaccional se requiere para un componente dado. En tiempo de ejecución el MTS lleva a cabo todo el manejo necesario automáticamente. Esto reduce significativamente el tiempo de desarrollo y facilita la reutilización de componentes.
Pool de conexiones a bases de datos	MTS genera un pool de conexiones a base de datos. Debido a que los componentes solo requieren de la conexión cuando están activos, el número total de conexiones requeridas para soportar los clientes sé decrementa, y así se incrementa el funcionamiento de la base de datos
Soporte para múltiples bases de datos y manejadores de recursos	Los componentes que corren en el MTS pueden accesar diversas bases de datos y otros recursos tales como colas de mensajes y aplicaciones mainframe simultáneamente, manteniendo la tolerancia a fallas. MTS maneja automáticamente las transacciones de tal forma que todas las bases de datos y otros recursos incluidos puedan aceptar o abortar la operación como una sola unidad.
Sin rastros en el cliente	Las aplicaciones MTS no requieren de librerías o ambiente de tiempo de ejecución en el cliente.

Aislamiento de procesos	Se pueden configurar los paquetes que contienen a los componentes para que se ejecuten en un espacio de memoria aislado y así prevenir que la falla de algún componente pueda perjudicar a otros paquetes.
Manejo automático de las instancias de los objetos	MTS crea instancias de los componentes en el servidor sólo cuando son llamadas, y reclama inmediatamente los recursos de memoria cuando el cliente finaliza su ejecución. A esta operación se le conoce como activación justo a tiempo (*Just-in-time activation.)*
Administrador de Propiedades Distribuidas	Esta herramienta simplifica la programación de los componentes que requieren guardar su estado entre sesiones o compartirla con otros componentes.
INTEGRACIÓN CON OTROS PRODUCTOS	
SQL Server	SQL Server provee un manejador de ODBC que es muy rápido y eficiente, basado en el Coordinador de transacciones distribuidas (DTC - Distributed Transaction Coordinator) que se integra perfectamente con el MTS.
Web Server	Internet Information Server utiliza los servicios de MTS tales como el manejo de transacciones. La integración con MTS permite que las páginas activas, que se encuentran ejecutándose en el IIS, puedan llamar a los componentes que se encuentran ejecutándose en el MTS y así tener acceso a bases de datos, aplicaciones mainframe y colas de mensajes. La integración con MTS también permite el aislamiento de procesos previniendo que fallas aisladas puedan afectar otras partes del sitio Web.
Integración con SNA Server	MTS puede utilizar SNA Server para incluir aplicaciones mainframe (por ejemplo aplicaciones corriendo en CICS e IMS) y manejar las transacciones de los datos en éstas. Las aplicaciones que hacen uso del MTS pueden actualizar una o más bases de datos, mandar y recibir mensajes e invocar a una aplicación mainframe que haga una actualización de datos, y todas las actualizaciones incluyendo aquellas sobre mainframe que se aceptan o se abortan como una sola unidad.
Soporte para servicios de agrupamiento	MTS soporta la operación sobre servidores Windows NT en grupos, permitiendo a los administradores configurar los paquetes del MTS para operaciones de *failover* y *fault-tolerant*.
Integración con la seguridad de Windows NT	MTS ofrece una seguridad basada en roles que se encuentra integrada con la seguridad de Windows NT. Los roles le permiten a los desarrolladores diseñar mecanismos de seguridad dentro de los componentes sin preocuparse de cómo serán implementados. Los administradores pueden mapear los roles a los usuarios al instalar el componente dentro del MTS.

Tabla 4.2 Resumen de la competitividad del MTS comparado con otras tecnologías

COMPARACIÓN	MTS	SUN´S ENTERPRISE JAVA BEANS	ORBS BASADOS EN CORBA
FACILIDAD DE USO			
Simplifica el desarrollo de aplicaciones de *n*-capas	Sí	Sí	Sí
Soporte para componentes COM	Sí	No	Muy limitado
Manejo simple de interfaces de programación	Sí	Sí	Sí
Soporte para paquetes de componentes	Sí	No	No
Administración por GUI	Sí	No	No
Utilería de instalación automatizada del cliente	Sí	No	No
FUNCIONALIDAD			
Manejo automático de transacciones	Sí	No	No
Pool de conexiones a bases de datos	Sí	A través de terceros	A través de terceros
Soporte para múltiples bases de datos y manejadores de recursos	Sí	A través de terceros	A través de terceros
Sin rastros en el cliente	Sí	No	No
Aislamiento de procesos	Sí	A través de terceros	No
Manejo automático de las instancias de los objetos	Sí	No	Manual
Administrador de Propiedades Distribuidas	Sí	A través de terceros	Sí
INTEGRACIÓN CON OTROS PRODUCTOS			
SQL Server	Sí	No	No
Web Server	Sí	Limitada	Limitada
Integración con SNA Server	Transaccional	No transaccional	No transaccional
Soporte para servicios de agrupamiento	Integrado	No	Limitada
Integración con la seguridad de Windows NT	Sí	No	Limitada

EL PAPEL DE IIS Y LOS SCRIPTS ASP

MTS, en combinación con *Internet Information Server (IIS)*, es ideal para la construcción de aplicaciones de tres capas con *front ends* basados en Web. IIS facilita el envío de páginas interactivas y ricas en contenido visual a los navegadores, y cuenta con un mecanismo de aplicación llamado *Active Server Pages (ASP's)*. Las ASPs son archivos de texto que contienen una combinación de comandos estándar HTML y de scripts basados en Visual Basic y JavaScript.

Los diseñadores de páginas Web utilizan ASPs para construir páginas Web estándares que contienen referencias a URLs con extensiones ASP. Por ejemplo, es práctica común el enviar formas de entrada a los clientes y designar en el URL de retorno una ASP que procese la respuesta. Cuando el cliente haya llenado la forma y enviado su respuesta, IIS busca si la URL contiene una extensión ASP y ejecuta el script ASP apropiado. El script en el archivo ASP puede realizar cálculos básicos, accesar bases de datos vía la interfaz ODBC, y construir una cadena de código HTML al vuelo y regresarla al navegador, y lo más importante es que las páginas ASP pueden llamar a componentes que corran bajo MTS.

Cuando una ASP llama a un componente MTS, la petición viene al MTS como si viniera de otro cliente más. El componente puede:

- ✓ Llevar a cabo cualquier cálculo requerido por la lógica de la aplicación
- ✓ Accesar una o más bases de datos vía ODBC (por ejemplo Oracle)
- ✓ Accesar aplicaciones mainframe, utilizando interfaces estándar de componentes COM, vía SNA Server
- ✓ Enviar y recibir mensajes vía *Message Queue Server (MSMQ)*
- ✓ Llamar a otros componentes que realicen otras partes de la lógica de la aplicación, permitiendo la reutilización de componentes. De hecho, las compañías pueden reducir el tiempo de desarrollo creando aplicaciones a partir de un conjunto de componentes preconstruidos o comprados.

Debido a que MTS ofrece el manejo automático de transacciones, si una condición de error ocurre dentro de un componente MTS o un script ASP, MTS deshará todos los cambios hechos a las bases de datos (incluyendo aquellos en mainframe, si es aplicable) y las colas de mensajes. Las transacciones MTS proporcionan una protección importante a la integridad de los datos requerida por los sistemas. MTS ofrece también, un ambiente sofisticado de componentes en tiempo de ejecución que puede manejar grandes cantidades de usuarios y ofrecer buenos tiempos de respuesta.

Con la combinación de IIS *Active Server Pages* e IIS, se pueden crear poderosas aplicaciones de comercio electrónico y de línea de negocios con interfaces interactivas y dinámicas basadas en Web. De hecho, mientras las ASPs pueden realizar por si solas cantidades significativas de cálculos y de accesos a bases de datos, Se recomienda el uso de las ASPs para el manejo de la presentación y al MTS para la lógica de la aplicación. Esta recomendación produce aplicaciones basadas en Web, construidas a partir de componentes reutilizable, que pueden soportar grandes números de usuarios, realizar cálculos complejos, y accesar a múltiples bases de datos y otros recursos tales como aplicaciones mainframe.

EL USO DE HERRAMIENTAS COMUNES PARA CONSTRUIR APLICACIONES DE TRES CAPAS

Las herramientas de desarrollo como Visual Basic y Visual J++ deben mucha de su popularidad a la facilidad que ofrecen para desarrollar aplicaciones de dos capas. El soporte de COM en dichas herramientas, hace fácil el desarrollo de aplicaciones utilizando una arquitectura modular basada en componentes.

Con MTS, los desarrolladores toman ventaja de las habilidades que tienen sobre las herramientas de desarrollo, para la construcción de aplicaciones sofisticadas de tres capas. Siguiendo unas simples reglas, los desarrolladores pueden generar componentes de lógica de aplicación para sus programas en la forma de "Dynamic Link Libraries" (DLLs) que soporten COM y que corran en la capa de negocios bajo el control de MTS. Entonces, cuando los componentes de la capa de presentación llamen a los componentes de la capa de negocios, *DCOM (Distributed Component Object Model)* guiará las peticiones hacia el MTS automáticamente.

APLICACIONES DISTRIBUIDAS PRODUCCIÓN-CALIDAD

Desde hace tiempo, las compañías se han dado cuenta de los beneficios de las herramientas de programación orientada a objetos y de la metodología de desarrollo. Inicialmente, los objetos tenían el mayor de su valor al ser usados en el cliente, tal podría ser el caso del desarrollo de la interfaz de usuario, debido a que las compañías carecían de un estándar para el uso fácil de los componentes en el servidor. Esta situación comenzó a cambiar con el advenimiento de la especificación CORBA *(Common Object Request Broker Architecture)* de OMG *(Object Management Group)*. La especificación CORBA establece una forma de distribuir objetos sobre la red en una variedad de servicios estándar, tales como transacciones y seguridad. La especificación fue respaldada por proveedores líderes en el mercado, y los usuarios fueron exaltados con la propuesta de objetos reutilizables que pudieran correr en cualquier parte de la red.

LOS RETOS DE LOS OBJETOS DISTRIBUIDOS

La propuesta de crear objetos una vez y utilizarlos en cualquier lugar de la red es una propuesta obligada, y no esta en duda que OMG haya despertado conciencia en la importancia de la propuesta de la especificación CORBA. Al mismo tiempo, la especificación CORBA tiene algunas debilidades. Por ejemplo, CORBA no es un producto; es una especificación. Los productos que cumplen con CORBA de diferentes proveedores, difícilmente pueden integrarse y si logran hacerlo es en forma muy básica.

A pesar de la premisa sobre el reutilización, los objetos CORBA carecen de la habilidad para modificar externamente propiedades tales como el comportamiento de la seguridad y las transacciones (es necesario codificar en el objeto o en su archivo de definición de interfases) lo que limita la reutilización en las aplicaciones reales.

Más aún, OMG asumió que especificando una colección de comportamientos y servicios de objetos, los proveedores cooperarían para proporcionar las piezas del rompecabezas que encajarían con los ORBs. En la práctica, esto no ha sucedido. Muchos de los servicios aún no han sido proporcionados del todo; aquellos que existen en forma de producto frecuentemente requieren de la integración substancial de los sistemas por los clientes.

DIFERENCIACIÓN COMPETITIVA

MTS define, de muchas formas, una nueva categoría de producto. MTS por si mismo proporciona una combinación de:

- ✓ Un modelo de desarrollo basado en componentes
- ✓ Una infraestructura robusta de servicios como el manejo de transacciones y la seguridad
- ✓ Encolamiento de mensajes, soporte para aplicaciones mainframe, e integración con el servidor Web.

Los dos competidores más cercanos de MTS son CORBA, *Object Request Brokers (ORBs) y Enterprise Java Beans de Sun Microsystems.* A continuación se presenta una análisis de estas dos especificaciones en comparación con MTS.

CORBA ORBs

- ✓ CORBA no es un producto, es una especificación resultado de diversos comités integrados por diferentes compañías con tecnologías competitivas de objetos distribuidos. Para ganar un acuerdo, los comités de OMG frecuentemente se vieron en la necesidad de definir especificaciones lo suficientemente vagas para poder satisfacer las necesidades de sus miembros. El resultado de esto son productos que cumplen con la especificación CORBA que provienen de diferentes proveedores y que en rara ocasión interoperan entre ellos. En otros casos, las especificaciones existen, pero no existe un proveedor que ofrezca un producto que satisfaga la especificación.
- ✓ Importantes inconvenientes de tiempo de ejecución como el proceso de creación y destrucción de instancias de objetos y la creación de un "pool" de recursos, no son resueltos por la especificación CORBA. Debido a esto los productos que cumplen con la especificación CORBA en ocasiones experimentan problemas significativos de escalabilidad.
- ✓ La especificación CORBA no detalla el soporte de características importantes orientadas a los componentes tales como la administración de la seguridad de los objetos y el requerimiento de transacciones que involucren tareas externas a la arquitectura. En la práctica, esto limita de forma severa la reutilización de los objetos.
- ✓ La mayoría de los proveedores de CORBA ORBs son pequeños y tienen recursos limitados para el desarrollo. En consecuencia, éstos se han enfocado en el desarrollo del núcleo de las capacidades de los ORBs y no en servicios más complejos como el *OTS (Object Transaction Service*), que permite la administración de las transacciones.
- ✓ Para utilizar transacciones con ORBs, los desarrolladores tienen que comprar implementaciones de OTS a terceros para adicionar complejidad y soporte a cargas de trabajo. El acceso a las colas de mensajes y aplicaciones mainframe es también normalmente proporcionado por terceros.
- ✓ Los ORBs más populares ofrecen una administración de sistemas muy limitada.

La cuestión es que los CORBA ORBs ofrecen una forma simple y poderosa de construir aplicaciones distribuidas de objetos básicas, pero no proporcionan la infraestructura requerida para la implementación a un nivel Producción-Calidad.

ENTERPRISE JAVA BEANS

Sun Microsystems definió *Enterprise Java Beans* como una iniciativa que permite dar solución a muchas de las carencias de CORBA. Como CORBA, *Enterprise Java Beans* promete a los desarrolladores el permitirles la construcción de componentes reutilizables que corran en cualquier lugar de la red. Como CORBA, la estrategia de *Enterprise Java Beans* tiene deficiencias significativas:

Enterprise Java Beans no es un producto, es una especificación que es el resultado de la entrada de muchas compañías con intereses competitivos.

Inconvenientes programáticos como el "pool" de conexiones a bases de datos no son solucionados directamente por la especificación de *Enterprise Java Beans.*

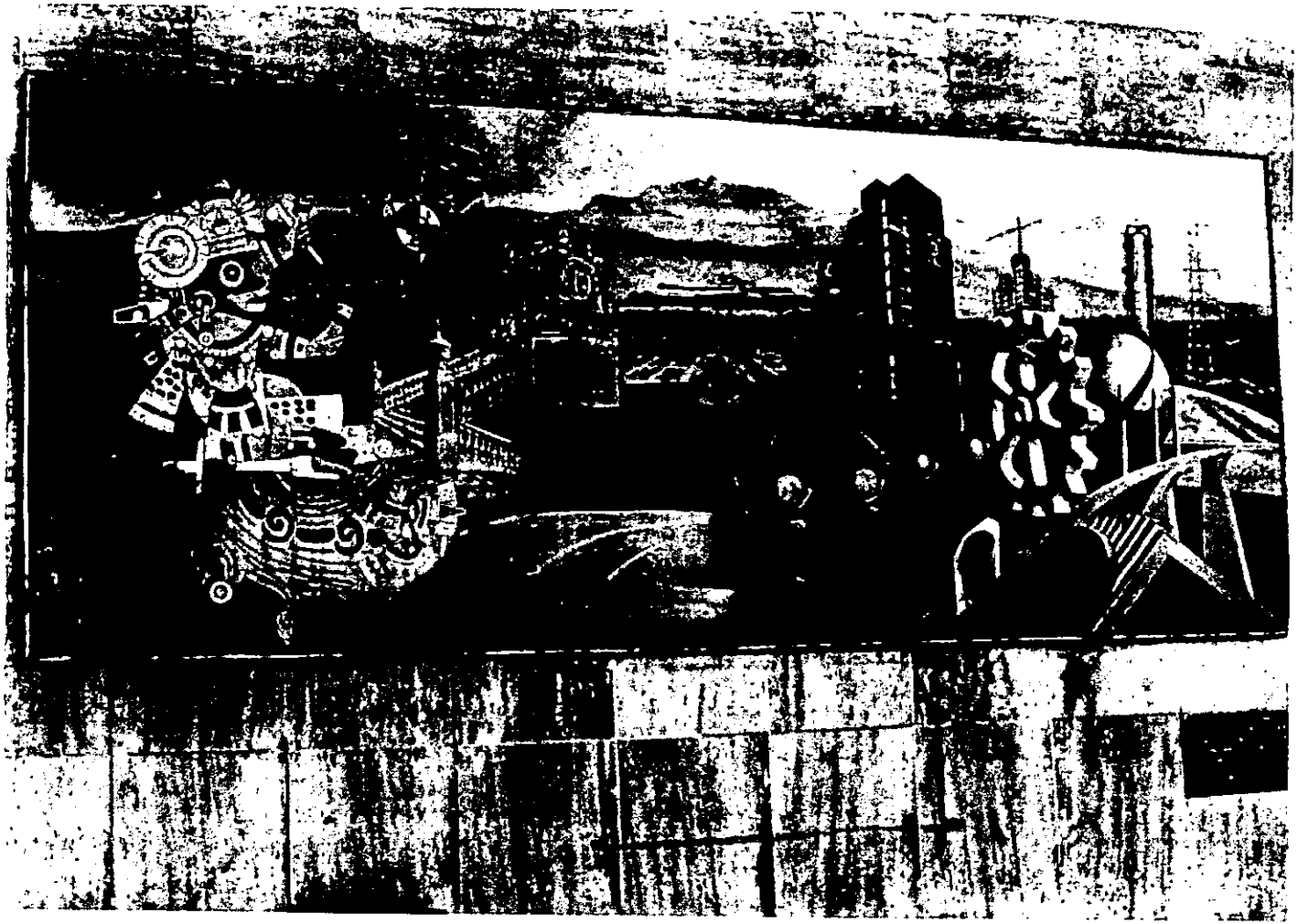
La especificación *Enterprise Java Beans* no detalla el soporte de características importantes orientadas a los objetos tales como la administración de la seguridad de los objetos y el requerimiento de transacciones que involucren tareas externas a la arquitectura.

Existe la necesidad de ensamblar varias tecnologías de diferentes proveedores y hacer integración de sistemas, para incorporar características importantes tales como el manejo de transacciones y el encolamiento de mensajes, dentro de las aplicaciones.

De hecho, hasta que Sun Microsystems no haga un gran esfuerzo para disponer más información sobre la estrategia *Enterprise Java Beans*, será difícil para los proveedores desarrollar productos que cumplan con la especificación y para los consumidores el contemplar a *Enterprise Java Beans* como una opción.

Para concluir este capítulo, cabe mencionar que históricamente existía un problema con la visión sobre los servicios centralizados. Se carecía de una herramienta y una infraestructura que permitiera construir e implementar componentes que se ejecutarán en un servidor. Las aplicaciones basadas en servidores necesitan ser más confiables que las aplicaciones que se ejecutan en el cliente, debido al impacto causado por una falla en el servidor o una corrupción de datos que pueda afectar al negocio completo, y no a un solo usuario. Lo anterior demanda una infraestructura sofisticada, costosa y difícil de mantener. Se estima que el desarrollo de esta infraestructura consume entre el 30% y el 40% del desarrollo del proyecto, lo cual resulta definitivamente poco viable.

En respuesta a esta problemática la infraestructura de servicios que ofrece Windows DNA combina la flexibilidad y el bajo costo de las aplicaciones de escritorio con las características de procesamiento transaccional de misión crítica normalmente encontradas en los sistemas mainframe. Y, debido a que manejamos el estándar COM el desarrollo de las aplicaciones sobre esta infraestructura, está al alcance de una gran cantidad de desarrolladores sin necesidad de un entrenamiento especial.



CAPÍTULO CINCO

IMPLEMENTANDO LA CAPA DE DATOS

Una vez definidas las bases sobre las que funciona un DNS es turno de comenzar a describir el papel y funcionamiento de las diferentes capas que propone Windows DNA. La capa de base de datos es la encargada de almacenar la información que fluye por nuestro sistema.

El buen diseño de esta capa repercute en gran medida en el funcionamiento e implementación de las capas de negocios y de presentación, si se hace un buen diseño de la base de datos, la programación y el rendimiento de la aplicación se verán beneficiados, por el contrario, un mal diseño puede traer consecuencias no deseadas y en ocasiones quizás sea necesario rediseñar la base de datos.

En el esquema tradicional Cliente/Servidor, un rediseño de base de datos implica la reprogramación de la mayor parte de la aplicación que tenga que ver con acceso a datos, en el esquema de Aplicaciones Distribuidas, un rediseño de base de datos solamente exige la reprogramación de los objetos que hacen acceso a datos. Esto no significa que el diseño de la base de datos pierda importancia, aun con esta flexibilidad un buen diseño de base de datos es la parte fundamental de todo sistema.

El proceso de planeación, diseño y desarrollo de bases de datos puede estar caracterizado por dos tipos de diseño el diseño lógico y el diseño físico:

- ✓ Diseño Lógico. El diseño lógico incorpora las reglas de negocios y muestra como pueden ser implementadas. A nivel base de datos, el diseño lógico transforma los escenarios conceptuales de negocios en relaciones de entidades, y más aun en un esquema de base de datos. La normalización en esta etapa ayuda a proporcionar un esquema de base de datos bien estructurado.

- ✓ Diseño Físico. El diseño físico transforma el diseño lógico en tablas, registros e índices. Durante esta fase, la afinación del desempeño y la optimización pueden ayudar a validar el diseño lógico de la base de datos.

La transición del diseño de bases de datos durante estas fases refleja los pasos involucrados en el diseño de la aplicación como un todo.

Como en el proceso del diseño de la aplicación, el proceso del diseño de bases de datos es interactivo. El modelo lógico usado para crear una implementación física no esta necesariamente completo; puede ser necesario volver a trabajar este modelo basado en situaciones que resulten durante el proceso del diseño físico, tales como consideraciones del desempeño o inconvenientes en la implementación.

MODELO LÓGICO DE DATOS.

La determinación de la estructura lógica de la base de datos y las relaciones dentro de esa estructura antes de la implementación de la estructura física puede ayudar a reducir un proceso grande y complejo como lo es el de un sistema de nivel empresarial a un proceso simple y administrable. Teniendo un diseño lógico también hace más fácil la conceptualización de los datos.

Un diseño bien planeado de base de datos no solamente mejora el proceso de desarrollo de la aplicación, sino que asegura un desempeño eficiente y confiable, manejabilidad y extensibilidad. El modelo entidad-relación se usa para el diseño de bases de datos relacionales y la definición de la estructura lógica de la base de datos.

MODELO ENTIDAD-RELACIÓN

Un modelo entidad-relación identifica entidades de datos y sus relaciones. Muestra los elementos significativos de la base de datos a alto nivel, mientras que ignora los detalles de la implementación tales como la indexación, administración del almacenamiento y la normalización.

El proceso del modelo entidad-relación requiere de los siguiente pasos:

1. Identificar las entidades u objetos mediante el análisis de los requerimientos de negocios

2. Identificar las llaves primarias que identifican de forma única cada entidad.

3. Identificar los elementos de datos o atributos asociados con cada entidad.

4. Identificar las relaciones entre entidades incluyendo la llave primaria y las relaciones con llaves foráneas.

Cuando este proceso es completado, tenemos entonces un modelo entidad-relación que representa una vista lógica de las entidades de datos y sus relaciones. Este modelo puede ser usado más tarde para definir la base de datos física definitiva.

MAPEO DEL ESQUEMA DE BASES DE DATOS

Una vez creado el modelo entidad-relación para la base de datos, se puede empezar a mapear el diseño lógico al diseño físico con la definición de un esquema de base de datos.

Durante esta etapa se convierten las entidades y sus relaciones a las estructuras de datos y mapeos requeridos por la arquitectura específica. Para bases de datos relacionales, este paso incluye la definición de tablas, elementos de tablas, tipos de datos y llaves de bases de datos. Cuando el mapeo esta hecho, todas las entidades son asignadas a una estructura de datos, y todas las relaciones lógicas entre estas estructuras se encuentran ya definidas.

ENTIDADES

Una entidad representa un sujeto (tal como una persona, un lugar, un concepto o una cosa) que almacena información. Las entidades son piezas fundamentales en las bases de datos relacionales. En una base de datos relacional típica, una tabla representa una entidad.

La siguiente tabla describe los tres tipos de entidades.

TIPO DE ENTIDAD	DESCRIPCIÓN
Kernel	Una entidad es conocida como una entidad kernel debido a que no describe otras entidades y no es usada para definir una relación entre otras entidades. En esencia, describe el significado de la base de datos. Durante la parte inicial de la fase de modelado, estas entidades kernel son normalmente, las más obvias de identificar.
Asociativa	Una entidad asociativa es la que sirve para unir entidades kernel. Las entidades asociativas se hacen aparentes cuando se empiezan a considerar las relaciones entre las entidades kernel
Característica	Una entidad característica provee información adicional para una entidad kernel o asociativa existentes. Por ejemplo, una entidad de detalle puede contener información adicional acerca de una entidad kernel. Dado que la información puede cambiar con el tiempo, es mejor que almacenar esta información en una entidad separada.

Tabla 5.1 Tipos de Entidades

Las entidades son representadas normalmente como recuadros en un diagrama entidad-relación. Cada recuadro almacena texto que describe la entidad.

ATRIBUTOS DE LA ENTIDAD E IDENTIFICADORES

Una vez que las entidades han sido determinadas, los atributos y los identificadores de cada entidad son definidos.

Atributos de la entidad.

Los atributos son elementos de datos que definen las características de la entidad. Por ejemplo, una entidad "Cliente" es definida por atributos tales como nombre del cliente, dirección, teléfono, etc. Normalmente, se pueden determinar los atributos de la entidades examinando las características del mundo real. Sin embargo, en casos más difíciles se pueden requerir estudiar la definición de entidad, si es que existe. Después de determinar los atributos de la entidad, se puede añadir al diagrama entidad-relación.

Identificadores de entidad

Cada entidad debe ser identificada de forma única de entre otras instancias del mismo tipo de la entidad.

La siguiente tabla describe los dos tipos de identificadores de las entidades.

IDENTIFICADOR	DESCRIPCIÓN
Llave primaria	Una llave primaria es un atributo de la entidad que identifica de forma única cada ocurrencia de la entidad. Normalmente esta llave se convierte en la llave primaria física una vez que se encuentra definida la base de datos. Una llave primaria puede ser un atributo existente de la entidad, o puede ser necesario crear un nuevo atributo para tomarlo como llave primaria.
Llave foránea	Una llave foránea es un atributo de la entidad que es una llave primaria en otra entidad. Las llaves foráneas son usadas para implementar una relación uno a muchos o muchos a muchos.

Tabla 5.2. Tipos de Identificadores

DEFINICIÓN DE LAS RELACIONES ENTRE ENTIDADES

Juntos, entidades y atributos definen la información que pertenece a la base de datos. Las relaciones, otra parte del diseño lógico de la base de datos, definen como las entidades están asociadas unas con otras.

Una entidad asociativa relaciona dos entidades separadas. Una entidad característica es relacionada a una entidad específica, proveyendo más información acerca de la entidad.

TIPOS DE RELACIÓN ENTRE ENTIDADES

Existen tres tipos de relaciones entre entidades

- ✓ *uno a uno*
  En una relación uno a uno, una sola instancia de una entidad esta asociada con una sola instancia de otra entidad.

- ✓ *Uno a muchos*
  En una relación uno a muchos, una instancia de una entidad (conocida comúnmente como "Parent") es asociada a cero o más instancias de otra entidad (conocida comúnmente como "Child").

- ✓ *Muchos a muchos*
  En una relación muchos a muchos, varias instancias de una entidad son relacionadas a varias instancias de otra entidad.

DEFINICIÓN DEL ESQUEMA DE BASE DE DATOS

Una vez que se ha determinado el diagrama entidad-relación para el modelo lógico de datos, se requiere construir la estructura física de la base de datos a un formato de base de datos particular. Este formato es llamado esquema de base de datos. En una base de datos relacional, las siguientes reglas son usadas para mapear el modelo lógico a la base de datos física:

- ✓ La mayoría de las entidades en el modelo lógico se convierten en tablas
- ✓ Una instancia de una entidad es conocida como renglón o registro
- ✓ Un atributo de una entidad es conocido como columna o campo
- ✓ Los identificadores de las entidades se convierten en llaves primarias y en llaves foráneas
- ✓ Las relaciones de las entidades son mapeadas a relaciones entre tablas.

No existe necesariamente una correspondencia uno a uno entre las entidades y las tablas. La estructura de base de datos necesaria para representar propiamente el diagrama entidad relación puede involucrar más o menos tablas que las entidades.

TIPOS DE DATOS

Se debe asignar un tipo de dato para cada columna en la tabla. Los tipos de datos especifican las características del dato para una columna.

Algunos de los tipos de datos son:

- ✓ binary
- ✓ bit
- ✓ char
- ✓ date
- ✓ decimal
- ✓ float
- ✓ image
- ✓ int
- ✓ numeric
- ✓ text
- ✓ timestamp
- ✓ varchar

Además del tipo de dato, se debe determinar la longitud de la columna, la longitud debe permitir al usuario almacenar tantos datos como sean necesarios pero no desperdiciar espacio.

LLAVES PRIMARIAS

Los atributos de las entidades son definidos durante la fase de diseño del modelo lógico de datos. Estos atributos se mapean a llaves primarias en la base de datos.

Una llave primaria dentro de una tabla es una columna que identifica únicamente a un renglón. Para que una base de datos relacional funcione, cada registro dentro de la tabla debe ser identificado de forma única. Cuando un registro es adicionado, la columna de la llave primaria para ese registro debe ser actualizada con un valor único.

LLAVES FORÁNEAS

Una llave foránea es una columna en una tabla que es una llave primaria en otra tabla. Las llaves foráneas son usadas para implementar relaciones uno a muchos y muchos a muchos.

VALIDACIONES DE COLUMNAS

Una validación garantiza cierta conducta sobre un atributo. Algunos de las validaciones que se pueden utilizar en las columnas son las siguientes:

VALIDACIÓN	DESCRIPCIÓN
Not Null	Si una columna es especificada como Not Null, entonces un valor debe ser administrado a esta columna para que una operación de adición o actualización pueda ser completada.
Unique	Si una columna esta especificada como Unique, entonces los duplicados no son permitidos. Por ejemplo una llave primaria automáticamente tiene una validación unique.
Check	Si una columna tiene una validación de Check, entonces solo valores específicos son permitidos
Foreign Key	Si una columna es un llave foranea de otra tabla (como en una relación "parent-child"), entonces el registro "parent" no puede ser borrado si existen registros "child". Un registro "child" no puede ser añadido sin un "parent". A estas restricciones se les conoce como integridad referencial.

Tabla 5.3 Tipos de Validaciones

ATOMICIDAD DE COLUMNAS

En determinada situación puede ser necesario romper un atributo de una entidad en el modelo lógico, en múltiples columnas. Esta ruptura de atributos es considerada atomicidad de columnas. La atomicidad de las columnas hace más fácil la consulta y actualización de la información.

NORMALIZACIÓN

Crear un modelo de base de datos bien diseñado toma tiempo y experiencia. El modelado utilizando diagramas entidad-relación provee un esquema de base de datos que satisface la mayoría de los requerimientos. Sin embargo, normalmente se requiere de un mayor refinamiento.

La meta del diseño de una base de datos es estructurar relaciones entre datos para agilizar la búsqueda de la información necesaria sin la necesidad de duplicar la información. Los esquemas creados a partir de un modelo lógico y conceptual no necesariamente satisfacen este criterio. Un método para asegurar que el diseño de base de datos alcance tales requerimientos es la normalización de datos.

La normalización es el proceso de organización de datos para eliminar dependencias funcionales y crear una base de datos eficiente. La meta de la normalización es remover grupos de datos repetitivos, minimizar redundancia, eliminar dependencias y separar los atributos.

Un diseño óptimo de base de datos es el resultado de normalizar y desnormalizar hasta obtener algunas ganancias en desempeño. Esto ocurre normalmente durante el proceso de modelado.

Los beneficios de la normalización incluyen:

BENEFICIO	RAZÓN
Integridad de datos	No hay redundancia
Ordenamiento más rápido	Las tablas tienen comunas indexadas
Menos bloqueos	Las tablas tienden a ser pequeñas, los bloqueos afectan a menos datos
Consultas optimizadas	Las tablas normalizadas proveen "joins" rápidos y más eficientes.

Tabla 5.4. Beneficios de la Normalización

En forma general, un diseño lógico de base de datos debe satisfacer los siguientes tres niveles de normalización:

NIVEL DE NORMALIZACIÓN	DESCRIPCIÓN
Primera forma normal	No se pueden tener grupos de repetición o columnas multivalor
Segunda forma normal	Cada columna que no sea llave debe depender totalmente de la llave primaria. Si la llave primaria consta de más de una columna, una columna que no es llave no puede usar solo parte de la llave primaria
Tercera forma normal	Una columna que no es llave no debe depender de otra columna que no es llave.

Tabla 5.5. Niveles de Normalización

La cuarta forma normal (llamada Boyce Codd Normal Form) y la quinta forma normal existen, pero comúnmente no son usadas. Si estas formas normales son omitidas, el diseño puede quedar menos que perfecto y la funcionalidad no se vera afectada.

Un diseño basado en estas reglas proporciona un mayor número de tablas pequeñas. Esto reduce la redundancia de datos y decrementa el número de páginas requeridas en disco para almacenar los datos. Debido a que algunas relaciones entre tablas necesitan ser resueltas usando "joins" complejos, algunos RDBMS´s cuentan con un optimizador que esta diseñado par tomar ventaja de los diseños normalizados de bases de datos.

UNIVERSAL DATA ACCESS

La información que es usada para hacer decisiones de negocios cada día, es más que un lote de datos almacenados en bases de datos relacionales. Normalmente incluye datos personales en hojas de cálculo, aplicaciones de manejo de proyectos, correo electrónico, datos en mainframe tales como VSAM y AS/400, datos calculados al vuelo, y datos ISAM. En adición a todo esto, tenemos información disponible en los servicios de directorio, documentos en disco duro o en la red y la riqueza de datos en una variedad de formas en

Internet. Cada uno de estos tipos de datos necesitan ser tomados en cuenta cuando se realizan decisiones y cada uno esta en formatos diferentes, almacenados en una variedad de contenedores y accesados en una forma fundamentalmente diferente.

Existen tecnologías para el acceso a datos tales como ODBC, RDO, DAO, etc., las cuales han venido evolucionando en mejoras tanto de funcionalidad como de desempeño. Además existen API's que son usadas para accesar información de telefonía e-mail y sistemas de archivos.

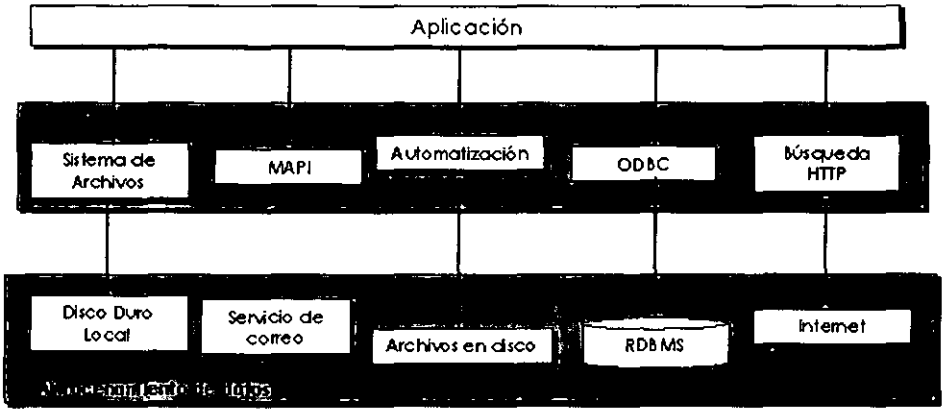


Figura 5.1. Debido a la gran variedad de repositorios de información se requiere de un acceso de una tecnología de acceso de datos para reunir la información necesaria.

La estrategia del acceso de datos de Microsoft conocida como UDA (Universal data Access) provee acceso a todos los tipos de datos a través de un modelo simple de acceso de datos.

UDA es un conjunto de interfaces comunes que pueden ser utilizadas para representar datos guardados en bases de datos relacionales, hojas de cálculo, contenedores de proyectos y documentos, VSAM, correo electrónico, o sistemas de archivos. Los almacenes de datos simplemente exponen un modelo de acceso de datos común hacia los datos.

La estrategia UDA, esta basada en OLE DB. OLE DB es un conjunto de interfaces C/C++ de bajo nivel diseñada para construir eficientemente componentes de bases de datos. Estas interfaces permiten un almacén individual de datos exponer su funcionalidad nativa fácilmente sin tener que hacer que los datos parezcan datos relacionales si no lo son, y permiten también, a los componentes de bases de datos genéricos y reusables aumentar la funcionalidad de simples proveedores cuando sea necesario. Debido a que son interfaces de bajo nivel que utilizan apuntadores, estructuras y manejo explícito de memoria para optimizar la forma en que los datos son expuestos y compartida entre los componentes, no son apropiados para ser llamados desde Visual Basic o Java. Haciendo una comparación se puede pensar en OLE DB como una tecnología de acceso a todos los repositorios de datos, ODBC los es para los datos SQL.

ActiveX

ActiveX Data Objects (ADO) es un "automation server" que expone un conjunto de objetos de acceso a bases de datos de alto nivel para la programación desde Visual Basic, Java o C++ que se encuentra un nivel arriba de OLE DB. Debido a que es una interfaz dual (no tiene apuntadores, estructuras, manejo explícito de memoria, etc.) es apropiado para ser llamado desde lenguajes de scripting tales como Visual Basic Scripting Edition (VBscript), y Microsoft Jscript. Los programadores de Active Server Page (ASP) usan los métodos de ADO para accesar y manipular datos almacenados en almacenes de datos OLE DB. Usando ASP, los programadores escriben código de lado del servidor para accesar los datos y construir una página HTML, la cual es entonces enviada al cliente. Toda la inteligencia y el código para accesar y manipular los datos existe en el servidor de Web. Centralizando esta lógica en el servidor de Web y reduciendo los componentes descargados al cliente es ideal para la mayoría de los escenarios de acceso a datos.

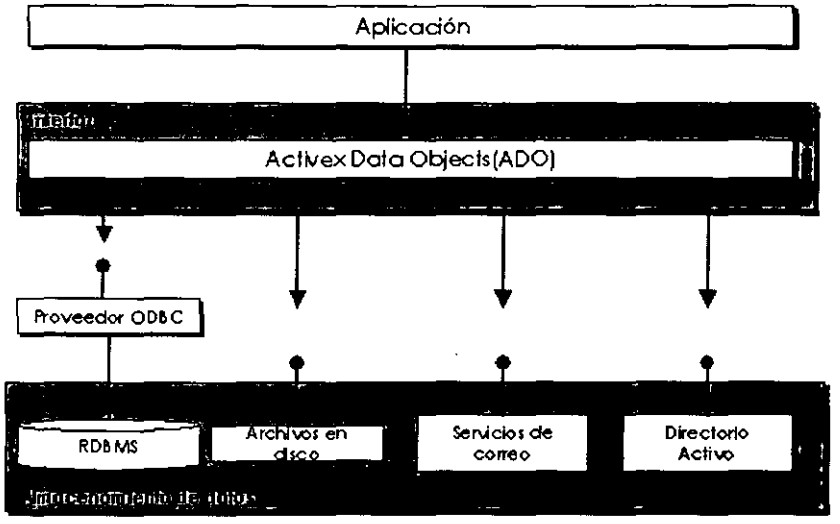


Figura 5.2. Acceso a datos con ADO.

Existe también un conjunto de componentes que proveen servicios remotos a ADO y OLE DB. Estos componentes son llamados Remote Data Services (RDS). RDS permite al cliente recuperar los datos tal cual en lugar de una página HTML textual. Esto es importante si el cliente requiere de usar datos en un formato diferente texto con formato. El cliente puede manipular datos locales a través de ADO o OLE DB directamente. RDS soporta "data bound controls" dentro del IE, que se encargan del manejo de datos remoto. El hecho es que estos servicios de datos remotos que son invocados, son invisibles para el consumidor de datos y el proveedor de los mismos.

OLE DB

OLE DB esta diseñado como una tecnología de componentes para alcanzar la meta de proveer acceso para todos los tipos de datos en un ambiente COM. Las fuentes de datos exponen interfaces COM que reflejan su funcionalidad, y de esta forma se pueden construir componentes comunes sobre estas interfaces para exponer modelos de datos más robustos. OLE DB logra conformarse como una tecnología de componentes ("componentization") debido a la identificación de características comunes entre los diferentes proveedores y servicios y a la definición de interfaces comunes que exponen estas características.

Un proveedor OLE DB expone interfaces OLE DB sobre los algunos tipos de datos. Ejemplos de proveedores incluyen desde un SQL DBMS hasta un archivo ISAM, archivos de texto o flujo de datos. Pero, al mismo tiempo, no es razonable esperar que todos los proveedores que exponen simples datos tabulares implementen un "query engine" poderoso. En vez de observar las diferencias entre los proveedores, OLE DB observa las similitudes y define interfaces comunes usadas para exponer estas similitudes. Como mínimo un proveedor OLE DB debe implementar y soportar los tipos de objetos y funcionalidad listados en la siguiente figura.

Con la implementación de este conjunto simple de objetos y un conjunto mínimo de conductas, un proveedor OLE DB es capaz de proveer un conjunto genérico de funcionalidad a los consumidores. La funcionalidad será probablemente escueta, pero un consumidor genérico estará habilitado para desplegar datos desde el repositorio de datos.

Es bueno contar con una forma común para representar datos, pero si los datos de los que se están hablando ofrecen funcionalidad adicional, tal como la posibilidad de ejecutar una consulta SQL sobre éstos, entonces ¿Cómo accesar esta funcionalidad extendida?. La especificación OLE DB, no sólo especifica las interfaces base y los objetos todos implementan, sino que también define extensiones comunes para exponer funcionalidad adicional que algunos repositorios de datos específicos puedan tener. El concepto clave para OLE DB es conformación como tecnología de componentes o "componentization". Microsoft divide las interfaces en grupos: algunos son soportados por todos los consumidores, y algunos son extensiones para áreas diferentes, tales como SQL, ISAM, etc.

Su conformación como tecnología de componentes permite a OLE DB exponer exactamente su funcionalidad nativa; así un componente separado puede ser añadido para implementar la funcionalidad adicional requerida por un consumidor. De esta forma se tiene un mundo de componentes cooperando, donde cada componente nativamente expone y consume interfaces OLE DB, y componentes individuales de bases de datos pueden ser añadidos de forma incremental a un repositorio de datos nativo para aumentar su funcionalidad.

Si se requiere que todo mundo accese a la información que se expone, entonces es lo indicado es construir un proveedor OLE DB.

La mayoría de los consumidores utilizarán ADO debido a que es más fácil de usar, otorga alto desempeño, y expone un conjunto rico en funcionalidad. Ciertos consumidores pueden decidir a utilizar interfaces OLE DB. OLE DB esta diseñado para hacer estos

componentes de bases de datos trabajen juntos, así que al final se puede tener control sobre cosas como la distribución y liberación de la memoria, etc.

ADO

ActiveX Data Objects (ADO) tiene similitudes con DAO y RDO debido a que se toman las mejores características de ambos modelos y las pone juntas en un modelo de programación común. Los puntos más importantes que podemos resaltar son:

- ✓ ADO es fácil de usar
- ✓ Independencia de lenguaje
- ✓ Provee interfaces extensibles para acceso programático para todos los tipos de datos

Dado que ADO consume interfaces OLE DB, cuyo único propósito es exponer la funcionalidad nativa desde diversos tipos de repositorios de datos, se pueden accesar todos los tipos de datos a través de ADO. ADO fue diseñado como un conjunto de interfaces que pueden ser comunes para cualquier librería de acceso de datos de alto nivel ofrece un modelo de programación común.

La independencia de lenguaje significa que ADO es un "automation server" que soporta interfaces duales. Esto quiere decir que ADO soporta la interfaz *IdIspatch*, así como la interfaz VTBL, de esta forma las aplicaciones escritas en Visual Basic, C++ y Java pueden ligarse directamente a través de la interfaz VTBL y obtener un mejor desempeño. Los lenguajes scripting pueden usar los mismos objetos a través de Idispatch.

Todos los objetos en ADO pueden ser instanciados sin necesidad de crear un objeto parent, excepto por los objetos Error y Field. La jerarquía de los objetos encontrada en modelos previos como DAO y RDO se rompe en el modelo ADO para permitir mayor flexibilidad en el reuso de objetos en diferentes contextos. Por ejemplo, se puede crear un objeto Command, asociarlo a una conexión y posteriormente asociarlo a otra conexión diferente. Este enfoque permite crear objetos especializados (definidos en tiempo de diseño) y recordsets temporales no ligados.

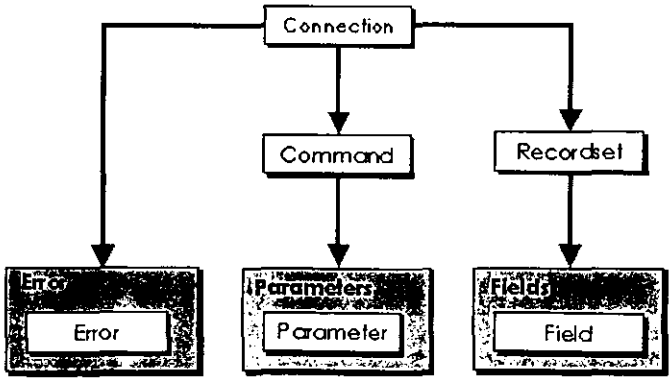


Figura 5.3. Modelo de objetos ADO

ADO como tecnología de acceso a datos es la opción que hemos encontrado más adecuada para para que sea utilizada en el desarrollo de los componentes de acceso a datos para nuestro DNS, ya que combina la flexibilidad, el performance y la facilidad de uso, que hacen de esta tecnología una buena opción.

Hemos llegado al final del capítulo durante el cual definimos el modelado de datos y el esquema de base de datos, y las tecnologías de acceso a datos. Los temas presentados son de gran relevancia para iniciar el desarrollo de un DNS. La capa de datos es la base sobre la cual se construye cualquier sistema informático y por lo tanto es de suma importancia el invertir tiempo en el análisis del esquema de base de datos. Las tecnologías de acceso a datos presentadas durante el capítulo permiten la comunicación de las otras capas que componen al DNS, y dan la oportunidad de guardar y extraer información de la base de datos. Aunque estas tecnologías cumplen con una función muy especifica, que exclusivamente tiene que ver con el acceso a datos, es importante mencionar que los objetos que hacen uso de estas tecnologías no forman parte de la capa de datos, estos objetos de acceso a datos forman parte de la capa de negocios y son precisamente el eslabón que conectan a la capa de datos con la capa de negocios, la cual abordaremos a continuación.

CAPÍTULO SEIS

LA CAPA DE NEGOCIOS

La capa de negocios es donde se desarrolla toda la lógica de la aplicación. Este capítulo nos introduce a muchos conceptos que tienen que ver con la lógica de negocios de una aplicación y define la forma en que ésta puede ser plasmada en componentes de software, así también como estos componentes de software se montaran dentro una estructura y donde un manejador cuidará de su correcta ejecución.

LOS NEGOCIOS HOY

Dos conceptos sobresalen en la conexión entre el mundo de los negocios y el mundo de la computación son: las reglas y los requerimientos de negocios.

Las reglas de negocios abarcan esas prácticas y políticas que definen la conducta de una corporación. Ampliamente utilizadas en las compañías, las reglas de negocio también generan las conductas únicas que resultan en una ventaja competitiva entre las compañías en el mercado.

Los requerimientos de negocios son imperativos, usualmente auto-impuestos, los cuelas son usados por las compañías para ayudarse a operar en un ambiente de negocios particular. Los requerimientos de negocios a menudo definen un punto de partida para los requerimientos de una aplicación y proveen una guía para el desarrollador. En términos prácticos, estos requerimientos de negocios son metas que los desarrolladores deben procurar lograr para implementar sus aplicaciones.

Juntos, reglas y requerimientos de negocios proveen un contexto para el desarrollo de aplicaciones que puedan ayudar a la compañía a lograr sus metas. Estas metas son únicas para cada compañía pero pueden ser generalizadas en las siguientes:

- ✓ Acceso oportuno y apropiado a la información
- ✓ Generar mejores decisiones de negocios
- ✓ Rápida respuesta a los cambios del mercado
- ✓ Mejora de la comunicación, entre trabajadores y con los clientes
- ✓ Animar la motivación entre los empleados
- ✓ Creación de un mejor retorno en la inversión

REGLAS DE NEGOCIO Y EL MERCADO

Las reglas de negocio son aquellos conjuntos de prácticas y políticas, algunas de éstas explícitas y otras implícitas, que definen como una compañía hace negocios con su mercado. El detalle de tal interacción es a menudo demasiado complejo. Para mejorar los negocios es necesario llevar a cabo un flujo constante de retroalimentación, para ello se deben suscitar las siguientes acciones:

1. Obtener la cantidad pertinente de datos del mercado como sea posible
2. Condensar, consolidar y priorizar estos datos en información útil
3. Analizar y priorizar la información acorde a la utilidad para la empresa
4. Crear un plan de acción que use esta información para llevar a cabo las metas
5. Llevar a cabo el plan

Esta actividad es continua y evoluciona para alcanzar las necesidades del mercado. La simplificación de la secuencia anterior asume que la compañía es de "una mente", la cual no siempre es verdad excepto para las compañías más pequeñas. Así, la compañía debe tener las siguientes características internas:

- ✓ Comunicación precisa y suficiente entre las partes involucradas
- ✓ Entendimiento consistente de las metas dentro de la compañía
- ✓ Una sola persona que sea responsable de cada plan.

REGLAS DE NEGOCIOS Y COMPUTACIÓN

Los sistemas de cómputo juegan roles importantes en la realización de las metas de las reglas de negocios y los beneficios de una interacción de la compañía con el mercado.

Los sistemas de cómputo proveen la habilidad para:

- ✓ Almacenar información procesada y datos del mercado
- ✓ Encapsular las reglas de negocios en software
- ✓ Cambiar rápida y fácilmente las reglas para responder a los cambios del mercado
- ✓ Actuar de forma automática sobre las reglas

De lo establecido anteriormente podemos decir que las reglas y los requerimientos de negocios son la base sobre la cual se cimienta el funcionamiento de cualquier empresa, y que los sistemas informáticos son la herramienta que ayuda al almacenamiento, definición y ejecución de esas reglas. Esta estructura corresponde a la capa de negocios, y son los objetos de negocios el componente de software representativo de las reglas de negocios.

OBJETOS DE NEGOCIOS

El mundo que nos rodea esta hecho de objetos físicos y de conceptos. Para modelar o simular de forma efectiva el mundo real y los procesos humanos en el código que desarrollamos, es necesario contar con la herramienta de software que permita esta representación de los objetos del mundo real. Ejemplos de estos objetos pueden ser:

- ✓ Una transferencia electrónica
- ✓ Una cuenta contable
- ✓ Una póliza contable
- ✓ Una bien o un servicio

Todos estos, son objetos de la vida real, y la idea principal es poder crear representaciones de estos mismos objetos dentro de las aplicaciones. De esta forma la aplicación permite interactuar a estos objetos entre sí, tal y como sucedería en el mundo real. Por ejemplo una transferencia electrónica puede crear una póliza contable que a su vez esta constituida de una o más cuentas contables que serán asentadas en la contabilidad.

El objetivo principal del desarrollo de aplicaciones es el de resolver algún problema o automatizar algún proceso. El tener un conjunto de objetos abstractos de software que representen objetos físicos, simplifica significativamente el proceso. Se pueden identificar

y diseñar los objetos de tal forma que estos proporcionen un modelo del negocio. Con la identificación inicial de estos objetos se comienza el modelado del proceso de negocio que se necesita automatizar o del problema a resolver.

Podemos entonces, definir un diagrama en el que se muestra de forma sencilla como es que los objetos se relacionan e interactúan unos con otros en el mundo real. Utilizando este diagrama podemos hacer que los objetos de software trabajen en la misma manera que lo harían los objetos del mundo real.

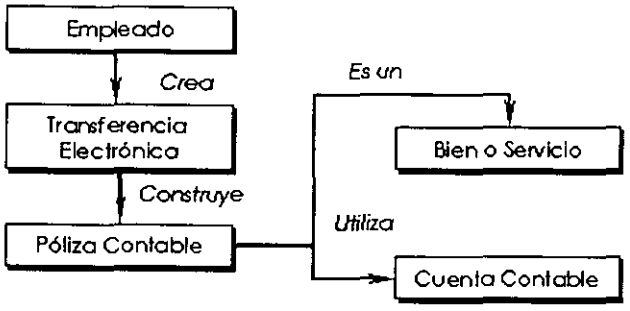


Figura 6.1. Diagrama que muestra la interacción de los objetos en el mundo real, de la misma forma se pueden crear objetos de software que interactúen de manera similar.

REUTILIZACIÓN DEL CÓDIGO

La mayoría de las aplicaciones que se desarrollan hoy en día son empezadas desde cero. Una y otra vez es necesario recrear muchos de los procesos y procedimientos utilizados en aplicaciones anteriores. Este procedimiento "cortar y pegar", se resume a copiar código de programas existentes y pegarlo y modificarlo para que cumpla con las especificaciones que define la nueva aplicación, lo que significa que el mismo código es copiado una y otra vez. La desventaja es que si es necesario cambiar algo en el código original, se tendrá que buscar y cambiar el código en todos los lugares en los que fue usado. Esto definitivamente dista por mucho de ser ideal.

En otras situaciones, la reutilización del código se lleva a cabo a través de librerías de código o inclusive rutinas precompiladas. Pero aún en estos casos la reutilización del código resulta difícil. Aunque esta opción ofrece ventajas sobre el "cortar y pegar", la desventaja aquí es que al cambiar la librería es necesario modificar y recompilar todas las aplicaciones que utilizan esta librería.

Así que el "cortar y pegar", ayuda, pero dista mucho de ser una buena solución, y la creación de librerías de código o rutinas precompiladas ayudan bastante, pero ambas técnicas aún dejan mucho que desear. Lo que se necesita es una solución que permita utilizar el código y que ofrezca la flexibilidad para hacer cambios basados en las necesidades particulares de una aplicación, o más aun, la posibilidad de modificar o extender el comportamiento. Por fortuna la programación orientada a objetos, ofrecen precisamente esta posibilidad.

¿QUÉ SON LOS COMPONENTES?

Los programas tradicionales se encuentran centrados alrededor de los procesos y procedimientos que el programa esta tratando de modelar. Los programas orientados a objetos, por otro lado, se encuentran centrados alrededor del mundo real de los objetos que el programa esta tratando de modelar. Una vez que estos objetos han sido identificados y modelados, se pueden escribir programas que utilicen estos objetos para simular los procesos y los procedimientos de una forma simple y sencilla.

Si basamos nuestras aplicaciones en objetos de negocios entonces es posible crear una colección de bloques que podemos usar una y otra vez para construir aplicaciones, y lo más importante es que es posible modificar y evolucionar los objetos con muy poco o nada de impacto sobre las aplicaciones existentes.

Los componentes forman parte de la tecnología orientada a objetos, la cual no es nada nueva y, de hecho, se ha venido utilizando desde hace muchos años, pero es ahora cuando comienza a convertirse en un concepto popular en los negocios. Estos se debe a muchas y diferentes razones, la más importante es la ansiedad de nuevas técnicas de desarrollo para facilitar la reutilización del código. Desafortunadamente, el simple hecho de crear objetos en código no es suficiente para maximizar su reutilización.

Por ejemplo si se tiene el código para un objeto en un programa, y necesitamos utilizar ese objeto en otra aplicación, lo que hacemos es copiar el código de ese objeto en la aplicación. En este momento hemos reutilizado el código, pero nada nuevo ha sucedido, porque si el código de ese objeto llegase a cambiar, entonces tendremos que buscar y cambiar el código en las diferentes aplicaciones que lo usan.

El problema es que tan pronto como existan dos o más copias de ese mismo código, no será sencillo mantenerlas sincronizadas en cuanto a su contenido.

Hasta este momento no hemos mas que utilizado nuevamente el simple "cortar y pegar", Es aquí donde el diseño orientado a componentes (component-oriented design) juega un papel muy importante. La meta es crear componentes construidos sobre tecnologías de objetos que proporcionen una mejor manera de alcanzar la reutilización de los objetos.

El término componente puede tener significados diferentes dependiendo del contexto en que se pregunte. Aún dentro de la comunidad OO (Object Oriented), existe un gran desacuerdo sobre el significado de este término. De tal forma que muchos pueden definir a un componente como un objeto de negocios grande y complejo.

De hecho, los términos "objeto" y "componente" son utilizados de igual manera, pero existe una diferencia importante entre ellos:

- ✓ Los objetos son creados a partir de clases. Las clases están construidas de código fuente que define los datos del objeto y las rutinas necesarias para manipular esos datos.
- ✓ Los componentes son código binario precompilado. Dado que los componentes son precompilados, son independientes del lenguaje en el cual fueron creados.

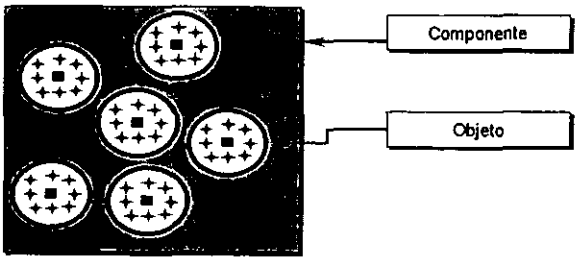


Figura 6.2. Componentes y objetos

Con los componentes binarios, podemos alcanzar un nuevo y completo nivel de reutilización en nuestras aplicaciones. De esta manera podemos poner el código en nuestro componente a diferencia de ponerlo en cada aplicación individualmente. Las aplicaciones pueden entonces utilizar el objeto directamente del componente.

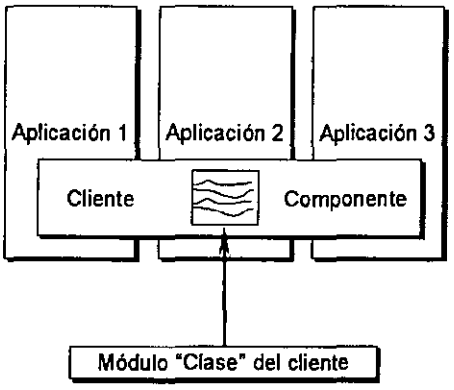


Figura 6.3 Componentes Binarios

Obtenemos la misma funcionalidad que cuando copiamos el código fuente en cada aplicación, pero la diferencia es que el código ahora se mantiene en un solo lugar, que es precisamente el componente.

OBJETOS Y CLASES

Para entender lo que es un objeto de negocio primero debemos entender que es un "objeto". Un objeto es una abstracción de una entidad o relación del mundo real representada en una porción de código. Un objeto se conforma de datos (información de estado) y de comportamientos (un conjunto de rutinas). Los programas pueden usar una "interfaz" para tener acceso a los datos y al comportamiento del objeto. Podemos representar a un objeto como se muestra en el siguiente diagrama:

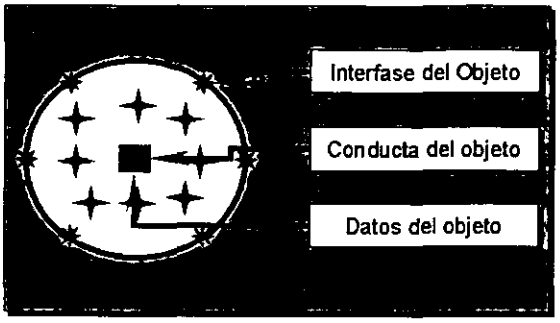


Figura 6.4 Objeto

Todos los objetos se encuentran definidos por una "clase". Una clase es esencialmente un molde o plantilla a partir de la cual un objeto es creado, lo que significa que podemos crear muchos objetos a partir de una sola clase. Cada objeto es referido como una "instancia" de la clase.

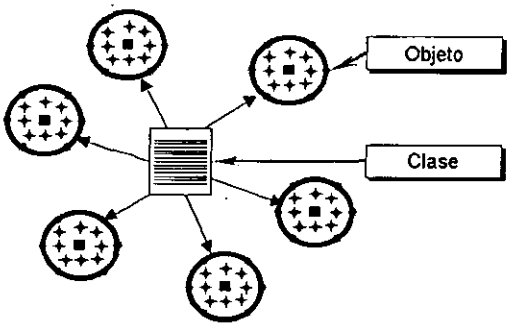


Figura 6.5 Objetos y Clases

Por ejemplo, consideremos la clase INSERT. De esta clase podemos crear instancias que representen a diferentes operaciones de ingreso en una compañía determinada.

En Visual Basic, las clases se definen utilizando los módulos clase, de esta forma podemos crear objetos basados en estos módulos clase.

LOS OBJETOS CONTIENEN DATOS

Todos los objetos contienen datos sobre ellos mismos. Por ejemplo, un objeto que represente a una POLIZA puede tener datos como el Total de la Póliza, la Compañía a la que pertenece, y el Periodo Contable. El objeto también puede saber si se encuentra Balanceado o Cancelado. Mas aún, consideremos a muchos objetos POLIZA, cada uno con su propio Total, su propia Compañía y Periodo, y cada uno de estos objetos pueden ser instancias de la clase POLIZA.

Los valores de los datos de los objetos se encuentran guardados en "variables de instancia". Cada instancia de una clase, tiene su propio conjunto de variables, que se

encuentra separado de las variables de cualquier otro objeto, incluso de aquellos que provienen de la misma clase. De esta forma dos objetos que provienen de la misma clase pueden tener su propio conjunto de variables separados uno del otro.

En Visual Basic, las variables de instancia de una clase se encuentran definidas a nivel módulo, dentro de un módulo clase. Por ejemplo:

```
Option explicit

Private str_Cia_Id as String
Private cur_ContaPol_TotalPol as Currency
Private str_Periodo_Id as String
```

Como en cualquier otro módulo, las variables declaradas a nivel módulo dentro de un módulo clase, pueden ser públicas (Public) o privadas (Private).

Las variables privadas son accesibles solamente al código que se encuentra dentro del módulo clase, y son la mejor opción para el manejo de los datos dentro de los objetos. Para que un cliente pueda manipular una variable privada, necesita llamar al código del objeto para hacer dicha manipulación, el cliente no puede hacer ninguna manipulación de forma directa. Esto proporciona un nivel de control muy bueno, dado que podemos implementar reglas o validaciones de negocios sobre cualquier dato que el cliente intente cambiar.

Las variables de instancia que son declaradas como públicas son accesibles al código dentro del objeto, pero también son accesibles al código del cliente. El uso de las variables de instancia públicas es una práctica mala, dado que cualquier cliente puede tener acceso a éstas y manipularlas sin tener que ser validado por el código del objeto.

Uno de los principios del diseño de objetos es la "encapsulación", que define que los objetos deben esconder su diseño y datos internos.

OBJETOS Y SUS COMPORTAMIENTOS

La idea principal de la creación de un objeto es que los programas puedan interactuar con él, de tal forma que si encapsulamos los datos de un objeto, entonces es necesario proporcionar el acceso a la información que el objeto guarda. Los objetos necesitan proporcionar una interfaz que permita a sus clientes tener acceso a los servicios que estos ofrecen.

En Visual Basic, la interfaz de un objeto puede ser una rutina Property, Sub o Function, o bien un evento declarado al utilizar la palabra reservada Event. Cualquiera de estas rutinas declaradas como públicas (Public) dentro de un módulo clase, se convierten en parte de la interfaz del objeto. En la realidad un objeto puede tener muchas interfaces.

PROPIEDADES

Una propiedad es un atributo que describe a un objeto. Es importante resaltar que las propiedades de un objeto no son lo mismo que sus datos. Los objetos normalmente contienen datos que no deben ser considerados como atributos. Por ejemplo, un objeto GRIDPOLIZA puede tener la propiedad DescripcionCuenta, pero el dato esencial puede ser el Id de la cuenta.

Las propiedades permiten a los diseñadores de los objetos seleccionar qué información de los objetos estará disponible a los clientes. También se puede seleccionar qué valores de una propiedad pueden ser cambiados por los clientes.

Para el ejemplo del objeto POLIZA, podemos proporcionar su Total:

```
Public Property Get Total() as Currency
       Dim int_Cont as Integer
       For int_Cont=0 to Grid.Rows-1
              Total = Total +  Grid.Columns("Cargo").CellValue(int_Cont)
       Next
End Property
```

Visual Basic ofrece tres diferentes formas de representar una propiedad, cuenta con rutinas del tipo Property Get, Property Let y Property Set, que son utilizadas para el manejo de las propiedades de un objeto. La rutina Property Get permite obtener el valor de una propiedad de un objeto, mientras que las dos rutinas que nos permiten dar valor a una propiedad son Property Let y Property Set, esta última es utilizada solamente para hacer referencia a propiedades que representan a un objeto.

MÉTODOS

Los objetos como sus análogos en la vida real, necesitan proporcionar servicios (o funciones) cuando éstos interactúan con su entorno. A través del uso de sus propios datos, o de los datos pasados a los métodos como parámetros, hacen la manipulación de la información para producir un resultado.

Los métodos son simples rutinas codificadas dentro de la clase para implementar los servicios que se necesiten proporcionar. Algunos métodos regresan valores o proporcionan información de regreso, este tipo de métodos se les conoce como "métodos interrogativos". Los métodos conocidos como "métodos imperativos" solamente proporcionan un servicio y no regresan ningún valor a la llamada a éstos.

En Visual Basic, los métodos se implementan por medio de las rutinas "Sub" y "Function" dentro de un módulo clase. Las rutinas Sub pueden aceptar parámetros pero no regresan ningún resultado al terminar su ejecución. Las rutinas Function también pueden aceptar parámetros y deben regresar un valor al terminar su ejecución.

La diferencia entre una rutina Function y una rutina Property Get es muy sutil. Ambas regresan un valor a la llamada a éstas y, de igual manera, ambas son subrutinas que regresan un valor y que se encuentran definidas dentro de un módulo clase.

La diferencia es menos programática que de diseño, es decir, se pueden crear objetos sin ninguna rutina Property, y solo utilizar métodos para todas las interacciones que se tengan con el objeto. Sin embargo, las rutinas Property son definitivamente atributos del objeto, mientras que las funciones pueden ser un atributo o un método. Si se implementan todos los atributos como rutinas Property Get, y todos los métodos interrogativos como rutinas Function, se puede crear un código más entendible.

Siguiendo con el ejemplo definido al principio del capítulo, se hace necesario que los programas que interactúen con el objeto POLIZA tengan que saber su detalle contable, así que necesitamos proporcionar la manera de hacerlo:

```
Public Property Get gridDetallePoliza() As String
   'Barre el grid
    'Vamos al principio para asegurar pasar por todos los registros
    Dim int_Cont As Integer
    SSDBGrid.MoveFirst
    For int_Cont = 0 To SSDBGrid.Rows - 1
        str_Mov = str_Mov & "#$#&" & _
        SSDBGrid.Columns("Cuenta").CellValue(SSDBGrid.GetBookmark(int_Cont))
        str_Mov = str_Mov & "#&" & _
        SSDBGrid.Columns("Concepto").CellValue(SSDBGrid.GetBookmark(int_Cont))
        str_Mov = str_Mov & "#&" & _
        SSDBGrid.Columns("Cargo").CellValue(SSDBGrid.GetBookmark(int_Cont))
        str_Mov = str_Mov & "#&" & _
        SSDBGrid.Columns("Abono").CellValue(SSDBGrid.GetBookmark(int_Cont))
    Next
    gridDetallePoliza = "4," & CStr(int_Cont) & str_Mov
End Property
```

La manera de hacer esto, resulta más útil cuando se adiciona código para validación de las reglas. Por ejemplo, no es posible obtener el detalle de una póliza si contablemente no es correcto. Sin impactar en el código del cliente, esto lo podemos manejar de la siguiente forma:

```
Public Property Get gridDetallePoliza() As String
Dim str_Mov
    'Verifica que el detalle tenga movimientos
    If SSDBGrid.Rows = 0 Then
        gridDetallePoliza = "ERROR"
        Exit Sub
    End If
    'Verifica que el detalle de la poliza este correcto
    If Not ChecaGrid Then
        gridDetallePoliza = "ERROR"
        Exit Sub
    End If
   'Barre el grid
    'Vamos al principio para asegurar pasar por todos los registros
    Dim int_Cont As Integer
    SSDBGrid.MoveFirst
    For int_Cont = 0 To SSDBGrid.Rows - 1
        str_Mov = str_Mov & "#$#&" & _
        SSDBGrid.Columns("Cuenta").CellValue(SSDBGrid.GetBookmark(int_Cont))
        str_Mov = str_Mov & "#&" & _
        SSDBGrid.Columns("Concepto").CellValue(SSDBGrid.GetBookmark(int_Cont))
        str_Mov = str_Mov & "#&" & _
        SSDBGrid.Columns("Cargo").CellValue(SSDBGrid.GetBookmark(int_Cont))
        str_Mov = str_Mov & "#&" & _
        SSDBGrid.Columns("Abono").CellValue(SSDBGrid.GetBookmark(int_Cont))
    Next
    gridDetallePoliza = "4," & CStr(int_Cont) & str_Mov
End Property
```

En este caso, se dispara una error al código del cliente cuando éste trata de llamar al método gridDetallePoliza cuando el detalle contable no es correcto. Lo realmente importante aquí, es que estos métodos están totalmente encapsulados dentro de su módulo clase. De esta forma podemos hacer los cambios necesarios al objeto sin impactar el código en el cliente.

EVENTOS

En términos de programación orientada a objetos, un objeto responde a eventos. En Visual Basic se tienen eventos un tanto diferentes, es decir, no sólo se pudo responder a ellos, sino también ocasionarlos. Por ejemplo, se puede utilizar la sentencia "Event" para declarar un evento y la sentencia "RaiseEvent" para disparar un evento.

Continuando con el ejemplo de la clase POLIZA, supongamos la necesidad de totalizar los movimientos al ser ingresados por el cliente. Y tal vez resulte necesario el notificar a los clientes que el total de la póliza ha cambiado, esto resulta sencillo al declarar la siguiente sentencia en la sección de declaraciones generales del módulo clase:

```
Public Event TotalizaCargoAbono(ByVal TotalCargo As String, ByVal TotalAbono As String)
```

Los eventos son siempre públicos, debido a que se deben disparar en cualquier cliente que haya hecho referencia al objeto. Con el evento declarado, podemos ahora, dispararlo con la sentencia RaiseEvent

```
Private Function ChecaCols() As Boolean
    'Barre el grid
    Dim cur_TotalCargos As Currency
    Dim cur_TotalAbonos As Currency
    Dim int_Cont As Integer

    SSDBGrid.MoveFirst
    For int_Cont = 0 To SSDBGrid.Rows - 1
       'Si todo esta bien lo va totalizando
       cur_TotalCargos = cur_TotalCargos +
       SSDBGrid.Columns("Cargo").CellValue(SSDBGrid.GetBookmark(int_Cont))
       cur_TotalAbonos = cur_TotalAbonos +
       SSDBGrid.Columns("Abono").CellValue(SSDBGrid.GetBookmark(int_Cont))
    Next

       RaiseEvent TotalizaCargoAbono(Format$(cur_TotalCargos, "Currency"),
       Format$(cur_TotalAbonos, "Currency"))
       ChecaCols = True
End Function
```

INTERFACES DE LOS COMPONENTES

Hemos dicho, básicamente, que un componente es un conjunto de objetos precompilados. Algunos de esos objetos pueden ser utilizados por programas cliente, mientras que otros pueden estar ocultos al mundo exterior, disponibles sólo para los objetos al interior del componente. En muchas maneras, un componente es como una pequeña aplicación, que pone su funcionalidad a disposición de otras aplicaciones.

Dado que un componente es en realidad un grupo de objetos, las interfaces del componente se encuentran constituidas de sus objetos públicos. Esto implica que existe una manera de indicar cuales de los objetos serán públicos y cuales serán privados. No existe una analogía real a este concepto dentro del diseño tradicional orientado a objetos.

El diseño orientado a objetos siempre asume que cualquier objeto puede interactuar con cualquier otro objeto dentro de las aplicaciones. El diseño orientado a componentes nos proporciona un mayor control dado que podemos escoger cuál de los objetos del componente estará disponible para ser utilizado por otras aplicaciones.

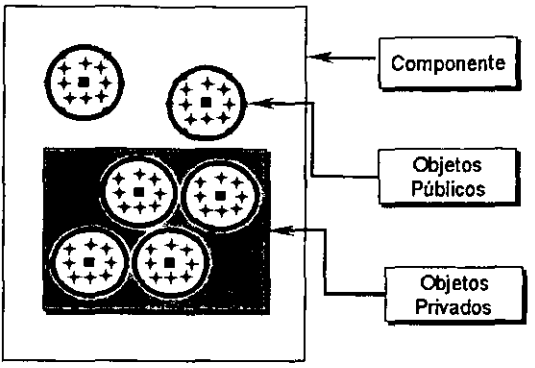


Figura 6.6. Interfaces públicas y privadas

Hasta este momento hemos visto como un objeto tiene una sola interfaz compuesta de subprocedimientos, funciones, propiedades y eventos, pero las cosas resultan ser todavía más complejas, debido a que los objetos pueden tener muchas y diferentes interfaces simultáneamente, aunque un cliente solo pueda utilizar una sola a la vez.

Si creamos un objeto a partir de una clase, este objeto tendrá por lo menos una interfaz, la interfaz que definimos en el módulo clase. En este caso, parece como si el cliente pueda tener acceso a todo el objeto, aunque en realidad se tiene acceso al objeto a través de la interfaz que se definió en la clase.

Con Visual Basic, un cliente nunca tiene acceso directo al objeto, solamente a alguna de sus interfaz. En algunas ocasiones una sola interfaz no es suficiente. El objeto puede representar mas de una sola entidad en el mundo real, y entonces es necesario poder modelar esta situación. Por ejemplo un objeto POLIZA en la vida real es también una PROVISIÓN CONTABLE. Para modelar esto, necesitamos que el objeto POLIZA actúe como el objeto PROVISION de vez en cuando.

Para poder hacerlo, solamente adicionamos la interfaz PROVISION a nuestro objeto. Entonces las aplicaciones cliente pueden utilizar al objeto a través de la interfaz POLIZA y su conjunto de propiedades y métodos, pero también pueden utilizar al objeto a través de la interfaz PROVISION con un conjunto de propiedades y métodos por separado. De esta forma las aplicaciones cliente pueden utilizar al objeto como mejor les convenga en un momento determinado.

Como hemos visto, cuando creamos una clase en Visual Basic, todo lo que se encuentra dentro del módulo clase comprende a la interfaz del objeto. Sin embargo, se puede utilizar la sentencia "Implements" para adicionar una interfaz más al objeto. La sentencia "Implements" requiere que se le proporcione el nombre de la clase con que esta trabajará.

Siguiendo con el ejemplo de la POLIZA, podemos adicionarle una interfaz a través de la clase PROVISION. Para hacer esto, necesitamos adicionar la sentencia Implements en la sección de declaraciones generales del módulo clase POLIZA:

```
Implements DoctoContable
```

Tan pronto adicionamos esta línea de código, Visual Basic requerirá que adicionemos código al módulo clase para implementar todas las propiedades y métodos de la clase DOCTOCONTABLE. Supongamos que la clase PROVISION tiene un método llamado TotalPasivo y una propiedad CuentaPasivo:

```
Public Function TotalPasivo() as Currency
End Function

Public Property Get CuentaPasivo() as String
End Property
```

La propiedad y el método pueden o no tener código en ellos. Como se muestra en este ejemplo, no existe código en ellos. Esto resulta no ser significativo para la clase POLIZA, debido a que es necesario agregar el código en la clase POLIZA.

CREANDO COMPONENTES DE NEGOCIOS EN VISUAL BASIC

Los objetos de negocios ofrecen reutilización de código y el modelado de los procesos de negocios ofrecen la reutilización en porciones de código de una manera entendible y manejable. Con la posibilidad de organizar y agrupar los objetos de negocios en componentes binarios precompilados, nos permite vislumbrar los beneficios de esta tecnología.

Visual Basic permite crear componentes de diferentes formas, así podemos organizar el negocio dentro de estos componentes. Haciendo uso de Visual Basic, se pueden crear servidores Activex, servidores In-Process (DLL) y servidores Out-of-Process (EXE). Además se pueden crear controles Activex (OCX) que son otra forma de componente, así cualquier tipo de estos componentes puede encerrar los objetos de negocios, cada uno con sus ventajas y desventajas.

SERVIDORES OUT-OF-PROCESS

Un servidor out-of-process es un componente, un programa por separado que contiene objetos. Este tipo de servidor siempre se ejecutará en su propio espacio de memoria, y por lo tanto se encuentra independiente de la ejecución de otros programas.

Los servidores out-of-process pueden ser tanto programas stand- alone que pueden ser utilizados por un solo usuario o que pueden servir a otros programas.

Existen dos formas principales de crear componentes out-of-process. Se puede contar con un servidor de tal forma que cada objeto se ejecute en su propio proceso, o se puede crear un servidor en el que todos los objetos se ejecuten en un solo proceso.

Existe un costo relacionado con que cada uno de los objetos se ejecuten en su propio proceso. La mayoría de las aplicaciones requieren que muchos objetos estén activos al mismo tiempo y si cada uno de los objetos se ejecuta en su propio proceso, entonces muchos proceso se encontrarán ejecutándose concurrentemente, lo que definitivamente decrementa el rendimiento de la estación de trabajo.

El sistema operativo Windows hace una verificación cada *quantum* para determinar que proceso será el próximo en la ejecución. Aun si un proceso se encuentra *"dormido"*, este forma parte de la lista de espera y de esta forma el sistema operativo tiene que realizar un trabajo extra, así el tener muchos procesos implica que el rendimiento se decremente.

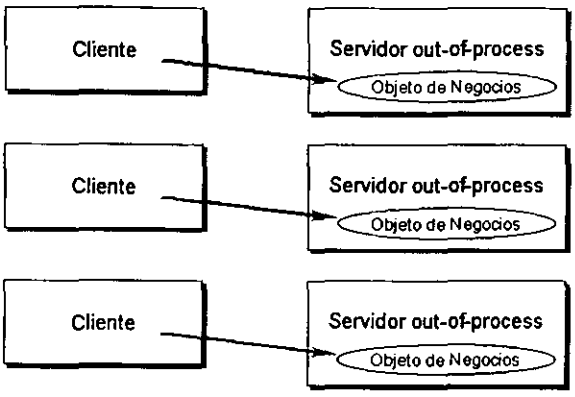


Figura 6.7. Servidor out-of-process SingleUse

Por otro lado, se puede crear un servidor en el que todos los objetos que se encuentran en él, se ejecuten en un mismo proceso. En este tipo de servidor, todos los objetos pueden trabajar juntos, compartiendo los recursos y la memoria. Muchos clientes pueden hacer uso de los objetos dentro del servidor y todos los objetos que sirven a todos esos clientes pueden comunicarse fácilmente y trabajar unos con otros dentro del mismo proceso.

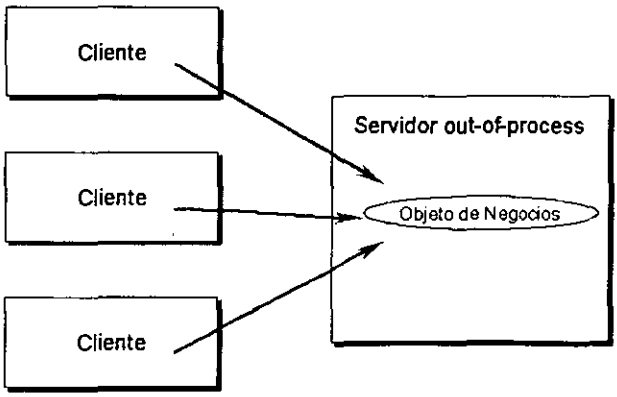


Figura 6.8 Servidor out-of-process MultiUseb

Todos los componentes en Windows utilizan la especificación de COM para poderse comunicar con otros componentes. Desafortunadamente, COM impone un alto consumo

en recursos para cada llamada que se hace entre un proceso y otro, y por lo tanto en cada referencia a una propiedad o en cada llamada a un método se consumen muchos recursos y esto hace que el rendimiento decrezca substancialmente.

SERVIDORES IN-PROCESS

Un servidor in-process es un componente en el cual los objetos se ejecutan dentro del proceso del cliente. Desde el punto de vista del cliente, este tiene su propia copia del componente y, por lo tanto una copia de todos los objetos. De hecho, el código del servidor puede ser compartido por cualquier número de clientes en la misma máquina, pero cada programa cliente tiene su propio conjunto de datos, dando la apariencia de que cada cliente tiene su propia copia del servidor.

Los servidores in-process evitan la sobrecarga de ejecutarse en su propio proceso, de esta manera, ahorran recursos y memoria para el sistema. Además, la comunicación entre el cliente y los objetos se hace dentro del mismo proceso, así se elimina el consumo de recursos impuesto por COM para la comunicación entre procesos.

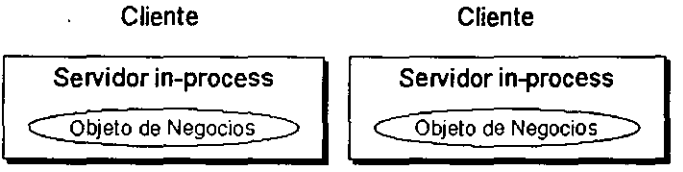


Figura 6.9 Servidor in-process

CONTROLES ACTIVEX

Este tipo de componentes, es utilizado para crear objetos visuales que formarán parte de una interfaz gráfica. Algunos controles ActiveX son invisibles, pero si hacemos un poco de análisis nos daremos cuenta que existe muy poca diferencia entre un ActiveX invisible y un servidor in-process.

Los objetos de negocios pueden estar contenidos en un control ActiveX invisible como lo estarían en un servidor in-process. Algunos desarrolladores encuentran que este tipo de controles son de fácil trabajo, ya que colocan literalmente el control dentro de la forma y hace uso de sus propiedades y métodos tal y como si estuvieran utilizando un control normal.

Los controles ActiveX son muy poderosos al ser utilizados en una aplicación Web. El explorador que soporte el uso de controles ActiveX, se encargara de descargar e instalar estos controles de forma automática.

Una desventaja de este tipo de componentes es que requieren que los programas clientes tengan por lo menos una forma que pueda contener al control. Mas que solo incluir una forma en el proyecto, es tener la forma cargada en memoria para poder tener acceso a los objetos dentro de ese control. Esto es de alguna forma menos eficiente que obtener los objetos directamente de un servidor in-process.

LOS COMPONENTES DE NEGOCIOS Y SU IMPLEMENTACIÓN EN LA CAPA DE NEGOCIOS

Una vez que hemos definido la importancia de las reglas de negocios y su implementación en objetos de negocios, así como qué son los objetos de negocios y como están constituidos, el siguiente paso es definir la forma en la que los objetos de negocios interactúan unos con otros para llegar a representar procesos. Para que los objetos de negocios puedan cumplir con su objetivo, requieren de una infraestructura que soporte su funcionamiento y que en conjunto aporten a la aplicación de las propiedades ACID en toda operación que se ejecute. La infraestructura que soporta y hace el manejo efectivo de los objetos de negocios es el MTS. Es en esta parte de la tesis es donde definiremos la implementación de los objetos de negocios en el MTS para que en conjunto constituyan la capa de negocios en la cual se maneja toda la lógica de la aplicación.

En las aplicaciones, los componentes de la lógica de negocios corren en servidores bajo el control de MTS, y son invocados por componentes propios de la capa de presentación vía HTTP, DCOM o RDS. La parte del cliente que invoca a los componentes de la capa de presentación, puede ser una mezcla de aplicaciones convencionales y scripts en ASPs ejecutados por el IIS.

Los componentes de la lógica negocios pueden tener acceso a un gran número de bases de datos relacionales y aplicaciones CICS e IMS. El acceso a las bases de datos y a los recursos es hecho a través de suministradores de recursos "Resource Dispensers" que dan servicios tales como el manejo de bloqueos y el "pool" de conexiones de forma automática. El MTS soporta también el manejo automático de las transacciones de tal forma que el acceso a los datos y a los recursos se lleva cabo en una base de "todo o nada".

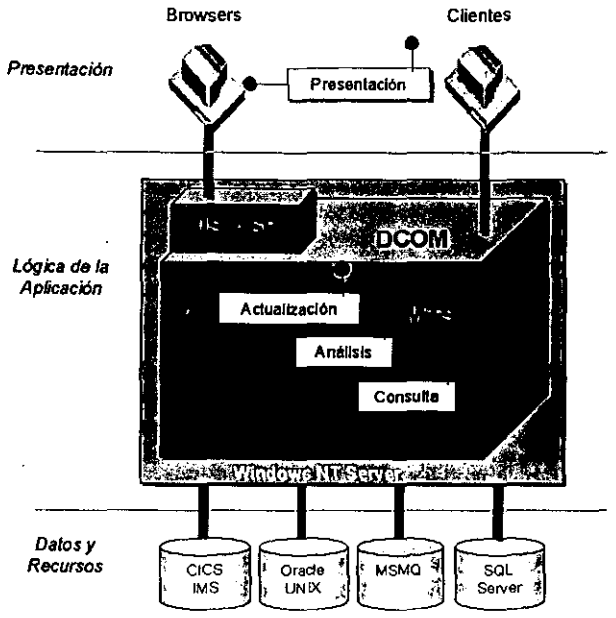


Figura 6.10 Tres capas de la arquitectura Windows DNA y sus componentes

Los desarrolladores pueden construir componentes propios de la capa de la lógica negocios con cualquier herramienta o lenguaje de programación que pueda generar componentes que cumplan con la especificación de COM. Para que los componentes puedan trabajar de forma efectiva dentro del ambiente del MTS, los desarrolladores deben observar ciertas reglas:

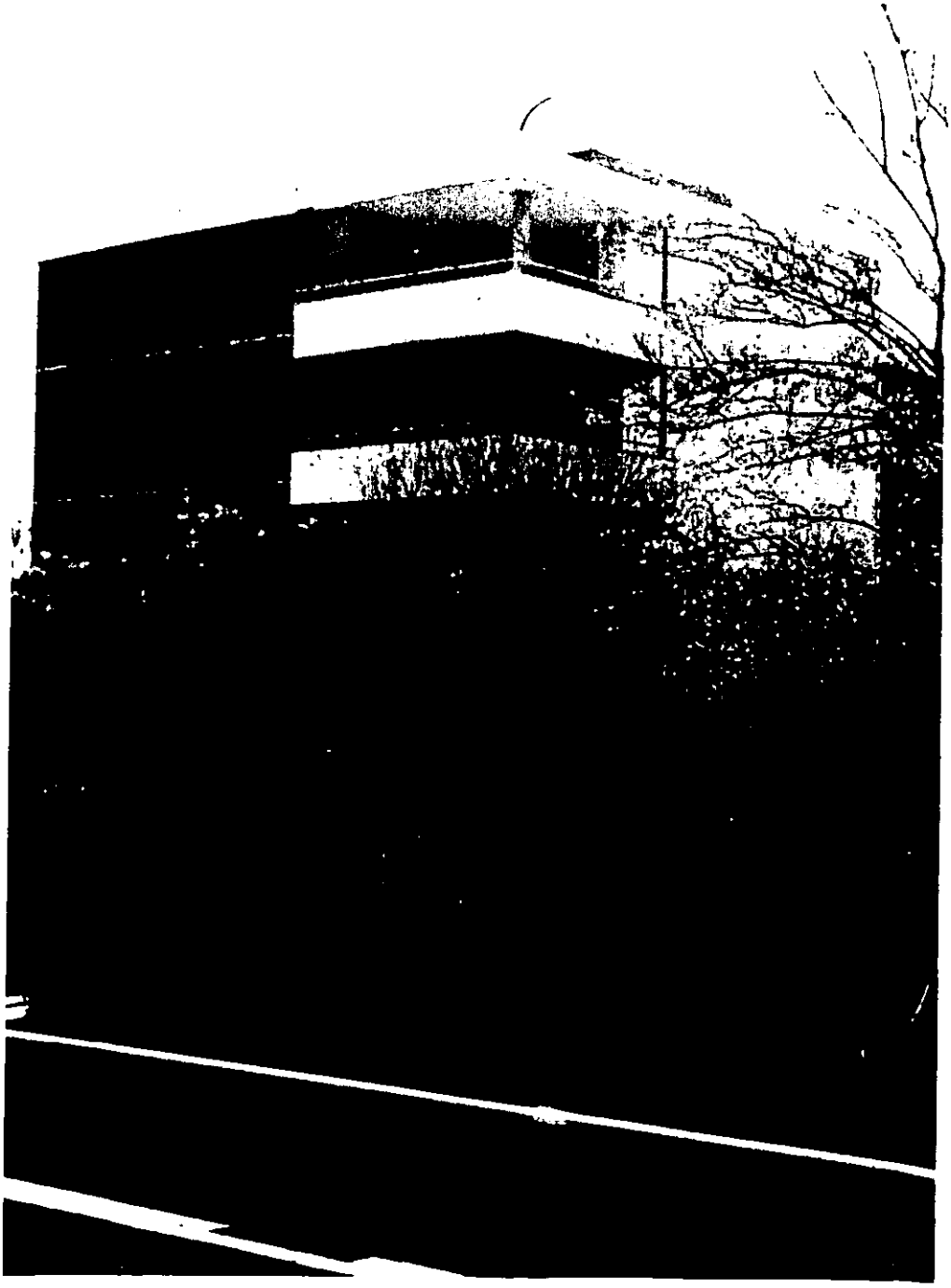
- ✓ Los componentes deben ser in-process DLL's
- ✓ Los componentes deben crear una referencia a su objeto de contexto (*MTS Context Object)* con la llamada a una API. La creación de esta referencia permite a este componente tomar ventaja de los servicios de MTS tales como el manejo de transacciones y la seguridad.
- ✓ No se debe guardar información de estado dentro de los componentes (por ejemplo en variables locales o globales) Los componentes que guardan información de estado son menos escalables, debido a que MTS no puede reciclar sus recursos cuando éstos finalizan su ejecución. La información de estado deberá ser guardada en bases de datos.
- ✓ Cuando un componente completa su ejecución satisfactoriamente, debe llamar al método *SetComplete* del MTS Context Object. Esto le dice al MTS que puede aceptar cualquier trabajo realizado por el componente cuando todos los componentes involucrados en la transacción finalicen su ejecución. El llamado al

método *SetComplete* también le dice al MTS que puede reciclar cualquier recurso que haya sido asignado al componente.

- ✓ Si un componente no puede completar satisfactoriamente, éste debe llamar al método *SetAbort* del MTS Context Object. Esto le dice al MTS que debe abortar la transacción en curso y deshacer los cambios hechos por los componentes involucrados en la transacción. El llamado al método *SetAbort* también le dice al MTS que puede reciclar cualquier recurso que haya sido asignado al componente.

Si los componentes siguen estas reglas, las aplicaciones pueden tomar ventaja de los beneficios del MTS tales como el mejoramiento de la escalabilidad, rendimiento, y la administración de los recursos sin tener que hacer programación extra.

Durante este capítulo definimos la capa de negocios, para lo cual fue necesario primero definir la importancia de las reglas de negocios dentro de una empresa y después implantar esta lógica en componentes de negocios que a su vez están constituidos por objetos de negocios. También trabajamos sobre la forma de crear objetos de negocios ayudados en un modelo de programación (cuya especificación se encuentra en el Apéndice B *Fundamentos de Diseño del Modelo de Programación*), estudiamos los tipos de componentes que se pueden crear con Visual Basic y terminamos definiendo los pasos a seguir para la implementación de los objetos de negocios dentro de la capa de negocios. Vale la pena mencionar que esta última parte es la esencia de la capa de negocios y ya que estamos parados al final vemos que es muy sencillo tomar los objetos de negocios y cargarlos en el MTS y dejar al MTS la responsabilidad del resultado y la consistencia de las operaciones que se realizan dentro de la aplicación. Si, en efecto es sencillo y la explicación de cómo implementar los objetos de negocios en el MTS puede resultar escueta, pero la realidad de las cosas es que no existe mayor complicación. Lo que sí debe quedar bien claro es que al final parece sencillo, pero para poder llegar a esta etapa, debimos haber pasado por una serie de etapas previas, todas ellas imprescindibles, que nos dan la pauta para decir en este momento que la implementación es sencilla.



CAPÍTULO SIETE

LA CAPA DE PRESENTACIÓN

Desde la definición de un sistema de n-capas se han venido explorando el cómo poder desarrollar aplicaciones de tal forma que la interfaz de usuario se encuentre separada de las demás capas que componen al sistema.

En capítulos anteriores se propuso una arquitectura donde la aplicación se encuentra compuesta de cuatro partes esenciales:

- ✓ La capa de presentación
- ✓ Los objetos de negocios UI-Centric
- ✓ Los objetos de negocios Data-Centric
- ✓ La capa de datos

Afortunadamente, los beneficios de dividir nuestra aplicación de ésta forma son claros. Con un esfuerzo mínimo se puede intercambiar la interfaz de usuario y de igual forma pasar de una base de datos a otra sin que los demás componentes de la aplicación se enteren de que algo dentro de la aplicación ha cambiado.

En una aplicación tradicional hecha en Visual Basic, por ejemplo, cualquier cambio en la interfaz de usuario o bien en la base de datos, repercutiría en una reestructuración del código bastante considerable. Con las técnicas se mencionaran a continuación ningún cambio en la interfaz de usuario podrá impactar al resto de la aplicación.

INTERFAZ HTML

En ésta parte se ilustrara que tan fácil es adicionar interfaces de usuario radicalmente diferentes. Se Podrá visualizar claramente que una interfaz tradicional hecha en Visual Basic (también conocida como interfaz Win32) o una de Office es muy diferente a la que un usuario experimenta utilizando un Web browser.

Mientras que la interfaz de Office o la Win32 son altamente interactivas, una interfaz de Web browser ésta orientada al manejo por *batch*. Es decir, el usuario introduce la información que se le pide en la pantalla y después oprime un botón, entonces la pantalla completa es procesada como una sola unidad. Además se proporcionan ligas a información útil y capacidades de navegación amigables, que normalmente no están disponibles en una interfaz tradicional de usuario.

HTML ha evolucionado para convertirse en DHTML (Dynamic HTML), que se encuentra disponible desde la versión 4.0 del Internet Explorer y en Comunicator 4 de Netscape. Con ésta evolución la flexibilidad tradicional de una interfaz de usuario ahora se encuentra disponible dentro de la ventana de un browser además del excelente manejo de texto e imágenes encontrados en el HTML ordinario.

Para implementar ésta interfaz, hacemos uso de las Active Server Pages (ASP). Las ASP proporcionan un ambiente de *scripting* del lado del servidor donde podemos interactuar con objetos pertenecientes al servidor de Web, utilizándolos para extraer información y construir dinámicamente las páginas Web "al vuelo". Éstas páginas Web

son documentos HTML o DHTML que llegan al cliente de la misma forma como se podría pensar se encuentran en el servidor Web, aunque como tales no existen.

Diseño de páginas para Internet

En la actualidad no existe una herramienta de *scripting* del lado del cliente verdaderamente universal. Ni la codificación en JavaScript o VBScript garantizan que correrá apropiadamente en todos los sistemas operativos y en todos los browsers. Lo que significa que no podemos utilizar cualquier *scripting* del lado del cliente si queremos asegurar que nuestra interfaz estará disponible a tantos clientes como sea posible.

El mismo caso tenemos para los controles o los componentes. Los controles ActiveX son muy poderosos, pero tienen una audiencia limitada debido a que son de tecnología Microsoft. A decir verdad, pocos browsers o sistemas operativos soportan los controles ActiveX, y nuevamente resulta necesario evitarlos si lo que se busca es abarcar el mayor número de clientes posible.

Para nuestro caso, que comprende el desarrollo de un sistema perfectamente orientado a un número de usuarios conocidos, como podría ser el caso de una Intranet o una Extranet, se puede definir un tipo de browser estándar, y con el tipo correcto expandir y abarcar todo el rango de tecnologías disponibles para éste, sin embargo no debemos pasar por alto que siempre estaremos interactuando con el HTML estándar.

Arquitectura de una Interfaz de Usuario

Como veremos más adelante podemos utilizar ASP para construir una interfaz de usuario donde el cliente mande o reciba HTML estándar. El usuario visualiza lo construido con HTML y proporciona información de regreso al servidor de Web utilizando formas HTML u otra técnica de intercambio de información válida.

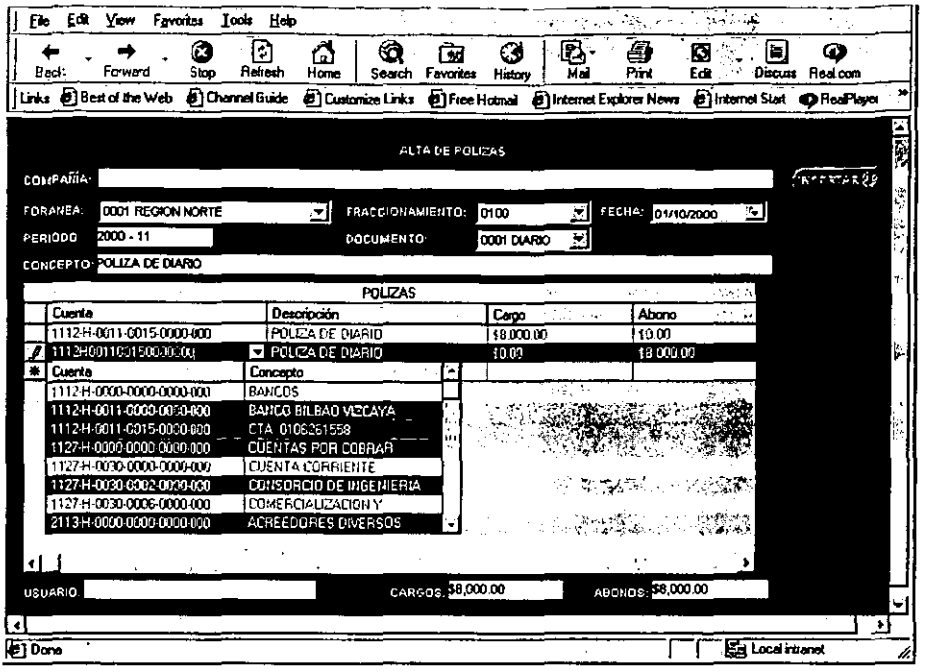


Figura 7.1

Los scripts ASP son responsables de utilizar la información proporcionada por el usuario para interactuar con los objetos de negocios y así generar código HTML que será enviado de regreso al browser. De ésta forma, el código ASP interactúa con los objetos de negocios de la misma manera que lo hace el código detrás de las formas en una aplicación de Visual Basic tradicional.

En una interfaz de Visual Basic tradicional es necesario adicionar código para permitir que los controles de las formas interactúen con los objetos de negocios. Los scripts en las ASP no son la excepción. Éstos permiten a las formas HTML interactuar con los objetos de negocios y enviar HTML de regreso al browser en repuesta a una actividad específica.

Los objetos de negocio como tales permanecen sin cambio. La comunicación entre éstos no es afectada por la interfaz de usuario, de manera que no importa si los objetos de negocio son usados por una interfaz de Visual Basic o un script ASP. En cada caso los objetos de negocio interactúan con otros objetos en el MTS exactamente de la misma forma.

ACTIVE SERVER PAGES

Los desarrolladores Web han utilizado diferentes técnicas para crear código del lado del servidor desde los inicios de ésta tecnología. Una de las primeras técnicas empleadas requería que el servidor Web reconociera ciertas extensiones como si fuesen scripts. El servidor Web entonces podía correr éste *script*, una vez que se le proporcionaban unos

argumentos incluidos en el URL, y mandar un *script* de regreso al browser. Éste intercambio de información se convirtió en un estándar *llamado Common Gateway Interface* (CGI).

Éste estándar, requería del uso de herramientas que soportaran la entrada y salida de argumentos o información al estilo del lenguaje C, lo que limitó de alguna forma ésta herramienta de desarrollo. En la mayoría de las ocasiones sólo los programas que trabajan en una línea de comando podían operar con CGI.

Con el tiempo comenzaron a surgir diferentes soluciones a las limitantes de comunicación a través de CGI. Microsoft creo ISAPI, que es muy similar en el concepto al CGI, pero que ejecuta los programas llamados dentro del mismo proceso del servidor para mejorar el rendimiento de éste. También se introdujo la noción de filtro ISAPI, que no es otra cosa que una DLL que intercepta cada petición por parte del browser y puede decidir que contenido regresar.

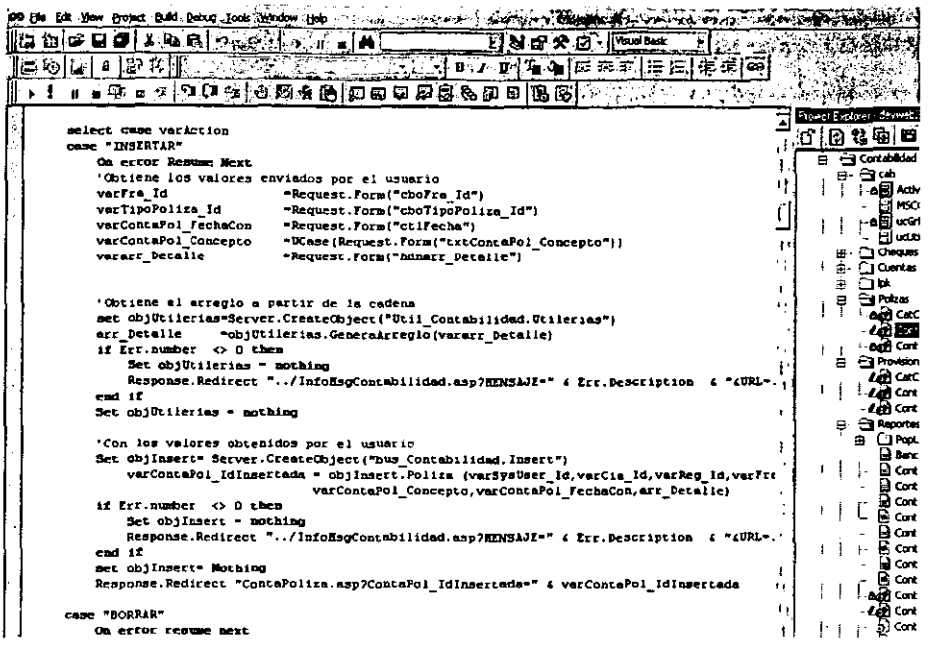
Mientras que ISAPI amplió el número de herramientas de desarrollo que se podían emplear del lado del servidor en el sistema operativo Windows, también sufrió algunas de las limitantes que presentaba CGI, una de las mas significativas es que era difícil de programar. Ninguna de éstas técnicas proporcionaba un ambiente de *scripting* real.

ASP es uno de los ambientes de desarrollo que intenta dar solución a los inconvenientes antes mencionados. ASP utiliza ISAPI para comunicarse con el servidor Web pero va más allá del mismo ISAPI, en las facilidades que proporciona a los scripts que corren dentro de su ambiente.

Un punto importante a mencionar es que la versión 2.0 de ASP tiene soporte transaccional sobre MTS 2.0, lo que significa que es posible codificar explícitamente el inicio y fin de una transacción dentro de una página ASP. Esto es interesante, pero en realidad no es relevante, ya que de acuerdo a la estructura propuesta para la aplicación, los componentes de negocio tienen el soporte transaccional implícito en los objetos que los conforman. Si dependemos de un *script* ASP para hacer el manejo transaccional de los componentes que son ejecutados por la página, y aceptar o deshacer la transacción dependiendo del éxito de todos los componentes involucrados, reduciríamos enormemente la flexibilidad de nuestra aplicación, debido a que la posibilidad de intercambiar las interfaces de usuario estaría condicionada a codificar en cada una de éstas el soporte transaccional.

SCRIPTS ASP

Antes de entrar de lleno a la revisión del ambiente ASP veamos a que nos referimos cuando hablamos de *script* ASP y como es que éstos trabajan. Para comenzar diremos que los scripts ASP son texto simple, y que los podemos crear utilizando desde el editor de texto más sencillo como puede ser el *Notepad*, hasta herramientas de diseño sofisticadas como Visual Interdev, HomeSite, HotDog, etc.



```
select case varAction
case "INSERTAR"
    On error Resume Next
    'Obtiene los valores enviados por el usuario
    varFra_Id              =Request.Form("cboFra_Id")
    varTipoPoliza_Id       =Request.Form("cboTipoPoliza_Id")
    varContaPol_FechaCon   =Request.Form("ctlFecha")
    varContaPol_Concepto   =UCase(Request.Form("txtContaPol_Concepto"))
    vararr_Detalle         =Request.Form("hdnarr_Detalle")


    'Obtiene el arreglo a partir de la cadena
    set objUtilerias=Server.CreateObject("Util_Contabilidad.Utilerias")
    arr_Detalle     =objUtilerias.GeneraArreglo(vararr_Detalle)
    if Err.number  <> 0 then
        Set objUtilerias = nothing
        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  & "&URL=.
    end if
    Set objUtilerias = nothing

    'Con los valores obtenidos por el usuario
    Set objInsert= Server.CreateObject("bus_Contabilidad.Insert")
        varContaPol_IdInsertada = objInsert.Poliza (varSysUser_Id,varCia_Id,varReg_Id,varFra
                                  varContaPol_Concepto,varContaPol_FechaCon,arr_Detalle)
    if Err.number  <> 0 then
        Set objInsert = nothing
        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  & "&URL=.
    end if
    set objInsert= Nothing
    Response.Redirect "ContaPoliza.asp?ContaPol_IdInsertada=" & varContaPol_IdInsertada

case "BORRAR"
    On error resume next
```

Figura 7.2. Visual Interdev como editor de páginas ASP

Para dar a conocer a nuestro servidor de Web que los archivos con extensión .asp son programas y no HTML simple, es necesario definir un directorio en el servidor que tengan permisos de ejecución. En el servidor Web se tiene por omisión un directorio llamado "C:\InetPub\wwwroot\" y es bajo éste directorio que podemos crear otros directorios en los cuales alojar los scripts ASP

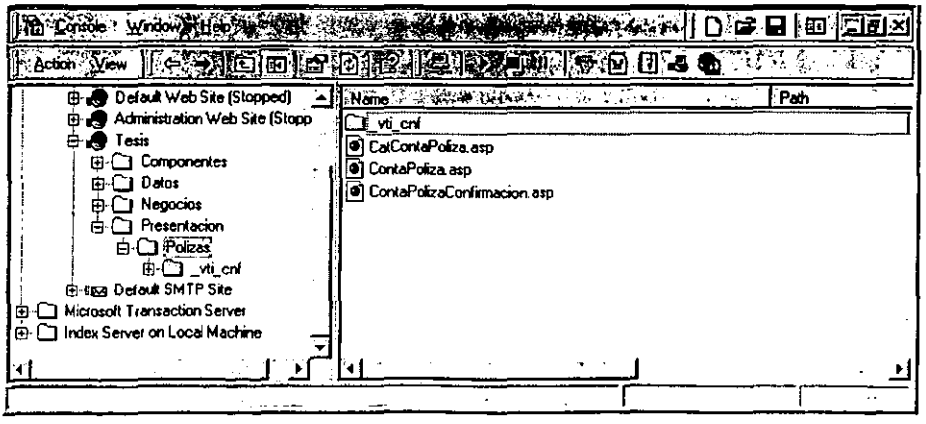


Figura 7.3. Una vez definido el sitio Web solo es necesario poner las páginas ASP dentro de los directorios creados para que sean ejecutadas por el engine de ASP

EL AMBIENTE ASP

Los scripts que se ejecutan en un ambiente ASP pueden ser hechos utilizando JavaScript o VBScript. Las ASP´s proporcionan a los desarrolladores un ambiente y un lenguaje poderosos con los cuales se pueden crear las páginas. Además proporciona soporte para el manejo de seguridad, así como el concepto de sesión (*session*). De hecho ASP ofrece cinco objetos principales en cada *script*, que exponen sus propiedades y sus métodos; éstos objetos son:

- ✓ *Application*
- ✓ *Request*
- ✓ *Response*
- ✓ *Server*
- ✓ *Session*

En ésta tesis no vamos ir al detalle de cada uno de éstos objetos, ya que por si mismo el tema de las ASP es muy extenso, sin embargo vamos resaltar los métodos de tres objetos que son el *Request*, Response y Server por ser los más utilizados en el desarrollo de páginas ASP´s.

EL OBJETO *REQUEST*

Comparando a ASP con otras tecnologías, una de las más grandes limitaciones de CGI e ISAPI es que no proveen una forma estándar o por lo menos fácil para que un *script* obtenga la información suministrada por el usuario o enviada como argumentos en el URL. El objeto *Request* es a respuesta a éste problema.

El código del *script* puede usar los métodos del objeto *Request* para obtener fácilmente cualquier dato provisto por el usuario, ya sea en una forma HTML o en el URL por sí mismo.

Request.QueryString

Una forma de enviar información a un *script* ASP es enviarla como parámetro en el URL. La sintaxis para éste tipo de intercambio de información en el URL es como se muestra a continuación.

```
http://servidor/directorio/página.asp?param1=valor1& param2=valor2
```

Del lado del servidor la obtención de éste valor enviado puede ser como la siguiente:

```
<%
Dim varParam1
varParam1=Request.QueryString("param1")
%>
```

Como se puede observar, la propiedad QueryString requiere que se indique el nombre del parámetro que se está buscando. Otra alternativa es usar un valor numérico para indicar la posición del parámetro.

Un punto que conviene hacer notar es que la variable `varParam1` no se declaró con un tipo de dato específico, ésto es debido a que en las ASP´s solo existe un tipo de dato, y éste es el Variant.

Request.Form

Las formas HTML son muy antiguas, al menos en términos de cronología de Internet. Éstas formas nos permiten crear pantallas de captura donde el usuario puede introducir valores en cajas de texto (*"inputs"*), seleccionar desde listas (*"multiple selects"*), cajas de chequeo (*"checkboxes"*) y trabajar con listas despleglables (*"selects"*).

El objeto Request hace muy fácil la obtención de valores introducidos por el usuario en cada uno de los controles de nuestra forma HTML. La colección de Form del objeto Request funciona de manera muy similar al método QueryString, requiere que le indiquemos cuáles son datos de los campos queremos obtener.

```
<%
Dim varName
VarName=Request.Form("txtName")
%>
```

EL OBJETO RESPONSE

El objeto Response es el opuesto al objeto *Request*. En lugar de recolectar la información proporcionada por el usuario, el objeto Response es el que usamos para mandar información al browser..

El objeto Response tiene el método *Write* que permite escribir directamente al browser.

```
<%
Response.Write "<HTML>"
Response.Write "<BODY>"
Response.Write "<P>Este el método Write del objeto Response </P>"
Response.Write "</BODY>"
Response.Write "</HTML>"
%>
```

Si éste *script* se ejecuta, el resultado será una página en código HTML y desde la perspectiva del browser no existe diferencia alguna entre una página originalmente codificada en HTML con ésta que produce el *script*.

EL OBJETO SERVER

Nuestro objetivo es utilizar scripts ASP para crear una interfaz de usuario desde la cual se invocarán a los objetos de negocios. Esto implica que nuestros scripts tendrán acceso a los objetos de negocios, así como nuestro código debe poder interactuar con las propiedades y métodos de los objetos.

Uno de los métodos más importantes del objeto Server es el método CreateObject. Al igual que en el método CreateObject de Visual Basic, solamente necesitamos proporcionar el nombre del servidor ActiveX y su clase, y éste método nos regresa una referencia al objeto.

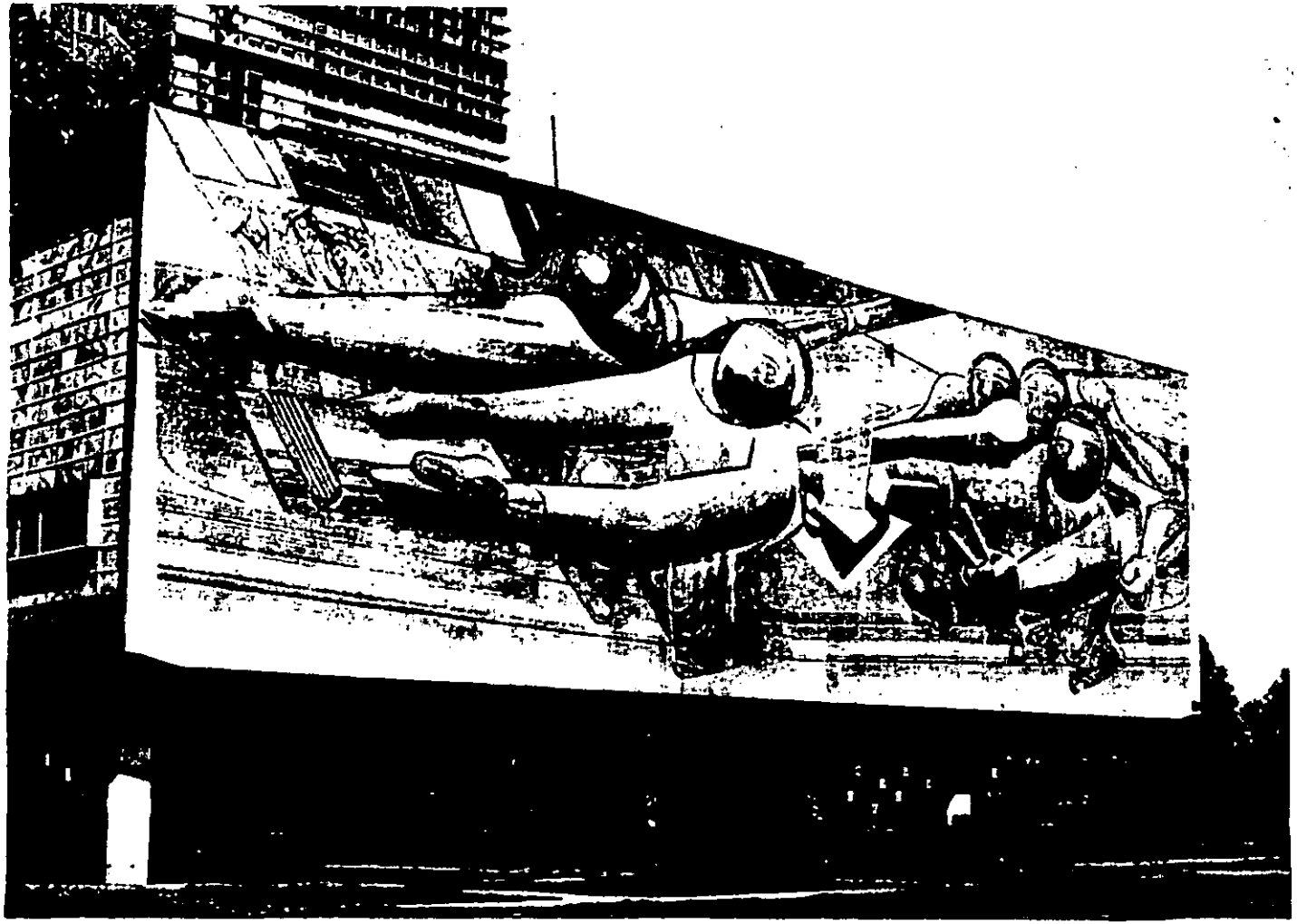
```
Set objNegocio= Server.CreateObject("bus_Negocios.Negocio")
```

Existen algunas diferencias entre el método del objeto Server y el método de Visual Basic. Por omisión ASP sólo nos permite crear instancias de objetos a partir de un servidor in-process, y no a partir de un servidor out-of-process. Afortunadamente, nuestros objetos de negocios están contenidos dentro de una DLL y por lo tanto no necesitamos preocuparnos, a parte de que como vimos en el capítulo de la Capa de Negocios, el MTS sólo admite servidores in-process.

Existen además de la tecnología de ASP´s, otras tecnologías disponibles para la implementación de una interfaz gráfica, que para nuestro caso constituye la capa del cliente. Éstas tecnologías van desde la interfaz Win32 creada propiamente en Visual Basic hasta las denominadas aplicaciones IIS y DHTML que tiene ventajas y desventajas unas con otras.

Para el caso de desarrollar un DNS hemos optado por la interfaz ASP´s que si bien no es la de más fácil desarrollo e implementación, si es la más flexible. En éste tipo de interfaz además del uso de las ASP´s es posible involucrar otras tecnologías como son VBScript, JavaScript, HTML, DHTML y obviamente el manejo de los objetos UI-Centric que aportan más poder a la interfaz.

Con éste capítulo llegamos a la conclusión de las tres capas más sobresalientes en el desarrollo de una aplicación distribuida, lo que resta en la tesis es asegurar el trabajo que se puede desarrollar con la concepción de un modelo de n-capas, por lo que a continuación abarcaremos el tema de la seguridad.



CAPÍTULO OCHO

SEGURIDAD

A lo largo de este capítulo estudiaremos la seguridad como un tema de gran importancia en el desarrollo de una aplicación y como un pilar para la implementación de un DNS. La seguridad es un tema muy amplio como amplia es la gama de posibilidades que existen para que un sistema sea "jaqueado". La seguridad en la implementación de un DNS bajo la arquitectura de Windows DNA comprende el manejo de la seguridad propia de cada uno de las diferentes partes o subsistemas que están involucrados en el funcionamiento del DNS.

Durante el desarrollo de la tesis hemos explicado las diferentes capas y partes que conforman un DNS y también podemos decir en este momento que el desarrollo del DNS bajo la arquitectura de Windows DNA implica la distribución de la aplicación misma en tres capas principales que son la capa de datos, la capa de negocios y la capa del cliente. Esta última encuentra en Internet, el vehículo ideal para la comunicación con los clientes; pero este vehículo es blanco de un gran número de ataques, debido a que la implementación del DNS esta en un servidor Web. A continuación se detallara la forma de reforzar la seguridad del DNS para cada una de las partes involucradas.

SEGURIDAD EN UN SITIO WEB

Debido a que *Internet Information Server (IIS)* y *Windows NT* son dos productos que se encuentran íntimamente integrados, es posible ofrecer el mismo nivel de seguridad que Windows NT ofrece. El beneficio de esto es la posibilidad de desarrollar sitios Web altamente integrados con el modelo de seguridad de Windows NT. El efecto colateral es que para poder desarrollar un sitio seguro es necesario primero entender el papel que IIS y ASP juegan dentro de Windows NT.

Cuando se implementa un sitio "seguro" es necesario considerar dos partes principales:

- ✓ Control de quién puede navegar el sitio
- ✓ Asegurar que la información intercambiada no pueda ser obtenida u observada sin autorización

Aunque estas dos etapas son completamente independientes una de la otra, pueden ser utilizadas en conjunto para poder proporcionar cualquier nivel de seguridad deseado.

El control de quien puede conectarse a un sitio Web y accesar sus archivos se implementa a través de una combinación de la seguridad de Windows NT e IIS.

El asegurar que la información transmitida sea segura es tarea de *Secured Socket Layer (SSL) y/o Private Comunication Technology (PTC)*.

SEGURIDAD EN WINDOWS NT

Lo más importante sobre seguridad en relación con Windows NT es que esta es "*pervasive*". Lo que significa que cualquier cosa que se haga en Windows NT involucra algún tipo de verificación. Si se intenta accesar un archivo, un chequeo de seguridad es llevado a cabo, lo mismo sucede si se trata de accesar a una estación de trabajo. Cuando

se usa Windows NT como sistema operativo de escritorio, estas implicaciones normalmente son invisibles. Esto es porque en la mayoría de los casos el acceso a la máquina es usando la cuenta de Administrador, y por consecuencia no existen problemas de permisos. En otras palabras, la cuenta de Administrador en un servidor Windows NT es el todopoderoso de esa máquina.

Debe notarse, sin embargo, que estos chequeos de seguridad se extienden a cualquier otro acceso en una máquina Windows NT. Por ejemplo, si otra máquina intenta accesar un archivo o un recurso, nuevamente un chequeo de seguridad tendrá lugar. Esto es especialmente importante considerando la función del IIS, que permite a los Web browsers accesar los archivos de una máquina Windows NT.

CUENTAS DE USUARIOS

Toda la seguridad en Windows NT se centraliza en las cuentas de usuarios. Dependiendo de sí se accesa a una LAN o no determina que cuenta de usuario de esta usando. Esto puede sonar un poco confuso, pero es importante hacer notar la diferencia entre ambos. En un escenario de una máquina "stand alone", todos los privilegios sobre esa máquina son determinados por la cuenta de usuario local. Por ejemplo, esto incluye todos los accesos a archivos, la posibilidad de instalar programas, y la posibilidad de ejecutarlos.

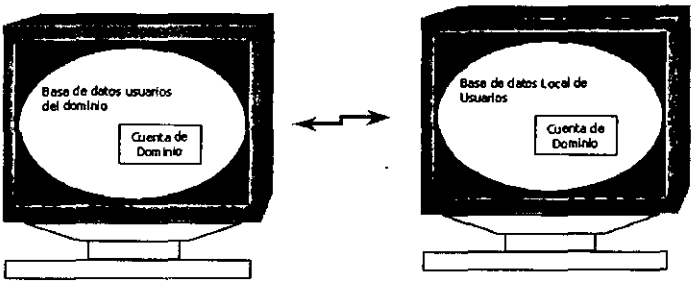


Figura 8.1 El Controlador de Dominio

En el diagrama de arriba se puede observar que cuando se trabaja en una red, las cuentas residen en dos lugares. Estas cuentas son almacenadas en una base de datos de usuarios. La base de datos de usuarios que es responsable de seguir el rastro a todos los usuarios sobre la red esta localizada en una máquina llamada Controlador de Dominio. Cada vez que se accesa a una máquina Windows NT es necesario identificarse como un usuario válido en una de las dos bases de datos. En un escenario "*stand alone*" es la base de datos local de la máquina, la encargada de validar el acceso a los usuarios, mientras que si el acceso es a una red, la base de datos del Controlador de Dominio se encuentra visible a todas las máquinas que hayan ingresado propiamente a la red, mientras que la base de una máquina que no esta en red solamente se encuentra visible a esa máquina en particular.

Todos los servidores Windows NT vienen con un conjunto de grupos predefinidos al momento de la instalación. Estos grupos se basan en un conjunto de roles comunes que se emplean cuando se trabaja en una máquina. Cada grupo tiene un nivel de privilegios y derechos basados en estos roles. Por ejemplo el grupo "Power User" tiene más derechos y privilegios que el grupo "Users". Por lo tanto los grupos Locales juegan dos roles. Desde

un punto de vista administrativo permiten al administrador organizar a sus usuarios en grupos, y al mismo tiempo les otorgan privilegios y derechos.

Los grupos de Dominio, difieren de los grupos Locales de forma considerable. Los grupos de Dominio sólo son un medio para organizar usuarios. La implicación de esto es muy sutil, lo que significa que haciendo a un usuario administrador del Dominio, no necesariamente resulta en administrador de las máquinas sobre la red. Para que esto suceda, cada máquina debe tener un grupo "Domain Admins" en su grupo "Administrators" local. Estos es, los únicos usuarios que pueden ser administradores de una máquina, son aquellos que se les ha otorgado directa o indirectamente privilegios de Administrador. Adicionando el grupo "Domain Admins" al grupo "Administrators" local, se esta otorgando indirectamente privilegios de administrador al grupo "Domain Admins" sobre la máquina en cuestión.

DERECHOS EN WINDOWS NT

El concepto básico de derechos es simple. Para poder ejercer una ación sobre un recurso, es necesario tener derechos para llevar a cabo esa acción. Los derechos en Windows NT son fáciles de entender, pero sin embargo es importante comprender que los derechos se encuentran en cada recurso que se intente accesar en una máquina. Esto significa que en lugar de almacenar información sobre que recursos puede accesar un usuario determinado, tal información es almacenada en el recurso en particular. Lo mismo aplica con los derechos de acceso al "*registry*" del sistema, los derechos de acceso al registry se encuentran en el mismo.

El hecho de que toda esta información es independiente de las cuentas de usuarios, resalta un hecho importante. Significa en esencia que el propósito principal de una cuenta de usuario es el de identificar al mismo. Lo que significa que al final todos los derechos de acceso apuntan a una cuenta de usuario. Por ejemplo, todos los permisos a todos los archivos apuntan a una cuenta de usuario, todas las políticas apuntan a una cuenta de usuario, y todos los privilegios al registry del sistema apuntan a una cuenta de usuario.

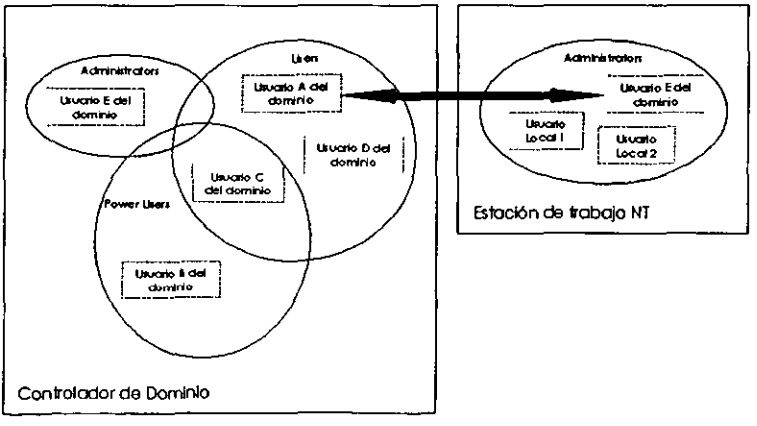


Figura 8.2 Grupos Locales vs Grupos de Dominio

Autenticación

La autenticación es un reto básico de autoridad, o en otras palabras, es el proceso de acceso a una máquina en una red o bien la revisión de las credenciales (nombre de usuario y contraseña) cuando se pretende hacer algo que requiere de ciertos privilegios. Después del acceso inicial a un servidor o a una estación de trabajo Windows NT toda esta autenticación toma lugar y es invisible al usuario. En un ambiente "*stand-alone*" al ingresar las credenciales que son cotejadas contra la base de datos local de la máquina. Después de ser validadas contra la de datos todas las acciones llevadas a cabo en la máquina son aprobadas bajo el contexto del usuario actual.

El acceso a un dominio es un tanto diferente. Cuando se accesa inicialmente a una máquina, se proporciona el nombre de usuario, la contraseña, y un nombre de dominio. La máquina en cuestión intenta contactar al controlador de dominio para el dominio especificado, y auténtico y casi en base al nombre de usuario y a la contraseña que se ha proporcionado pronto después de que esto se hecho, todos los derechos y privilegios a plantar han hay esta cuenta de varió en el dominio. Por lo tanto, como se ha visto anteriormente, esto significa que si se desea llevar a cabo alguna acción que involucre cierto nivel de administración, entonces la cuenta de usuario en el dominio debe pertenecer al grupo de administradores en la base de datos local.

ACCESO A RECURSOS EN WINDOWS NT

Los recursos de Windows NT son considerados objetos. Ejemplo de tales objetos puede ser una llave en el Registry, o un archivo en el sistema de archivos de Windows NT (NTFS). Todos los recursos de Windows NT tienen asociados privilegios a ellos. Tales privilegios se mantienen en las listas de control de acceso (*Access Control List o ACLs*). Las ACLs son listas que contienen identificadores únicos (SIDs) que representan a las cuentas de usuarios. Cada SID contiene archivos asociados a estos. La definición de tales privilegios en las ACLs puede variar y depende del objeto en cuestión.

Acceso a archivos

Cuando se trata de archivos que residen en una unidad formateada con NTFS, se cuenta con la capacidad de asignar privilegios a los archivos contenidos en un directorio o bien a un solo archivo individualmente. Este es una de los chequeos de seguridad que pasan desapercibidos al momento de accesar un archivo cuando se es administrador de la máquina local, en un ambiente de red estos chequeos resultan más aparentes.

Como se mencionó anteriormente, los privilegios sobre los archivos, se mantienen en las ACLs: Cada vez que se trata de accesar a un archivo, se hace referencia a esta lista para verificar si el SID del usuario se encuentra en la ACL y determinar los privilegios de acceso para ese usuario antes de permitirle el acceso.

Acceso al Registry del sistema

El acceso al registry del sistema es también administrado por la seguridad de Windows NT. Cada llave del registry tiene su propia ACL de forma similar a los archivos en NTFS. Mientras que tales privilegios no representan un problema para los usuarios interactivos, para los desarrolladores del *Internet Information Server (IIS)* si representa un problema, especialmente en el contexto de DCOM.

PROTOCOLOS AUTENTICADOS

Algunos protocolos para la comunicación entre máquinas sobre una red, requieren de autenticación. Dos ejemplos clásicos de protocolos que requieren y no requieren autenticación son *Named Pipes y sockets* TCP/IP. Named Pipes es un buen ejemplo de un protocolo autenticado porque es el protocolo de default que Microsoft SQL Server utiliza para comunicarse.

DERECHOS Y POLÍTICAS

Los derechos y las políticas se encuentran residentes en una locación. Cada derecho o política es entonces asociada con una lista de usuarios o grupos que tienen acceso a ese derecho o a esa política. El más obvio de estos derechos es el derecho a firmarse localmente o "Log on Locally". Este derecho permite a los usuarios accesar a la máquina y usarla interactivamente.

SEGURIDAD DE INTERNET INFORMATION SERVER (IIS)

La entender apropiadamente como es que interactuan Windows NT y IIS en cuanto a la seguridad es importante primero entender la verdadera naturaleza de IIS. Como lo hemos mencionado, IIS es simplemente una herramienta para proporcionar archivos a los browsers que los requisitan. En el entendido que cualquier interacción con una máquina NT necesita un chequeo de seguridad (autenticación), se puede suponer que cada petición de una página Web involucrará un proceso de autenticación. Conociendo esto, consideremos lo siguiente: si un usuario navega por un sitio Web contenido en una máquina Windows NT el servidor no tiene idea de la identidad del usuario, ¿Como entonces es que el usuario puede obtener los archivos que el solicita? Sabemos que es necesario que la autenticación se lleve a cabo, para este caso el usuario no podría obtener los archivos que solicita. La pregunta entonces es: ¿Cómo proporcionar contenido a un usuario que no se conoce? La respuesta a esta pregunta es el Usuario Anónimo (IIS Anonymous User)

USUARIO ANÓNIMO DE IIS

Cuando esta opción es habilitada en IIS, cualquier solicitud por una página que el IIS ejecute será bajo el contexto del Usuario Anónimo. IIS intentara accesar el archivo en cuestión como si el fuera el Usuario Anónimo. Una vez que el archivo es solicitado por el usuario anónimo, NT verificará la ACL de ese archivo para ver si el Usuario Anónimo tiene los privilegios de abrirlo. Si el Usuario Anónimo tiene suficientes privilegios, entonces el archivo es enviado.

Un Usuario Anónimo es dado de alta por default durante la instalación del IIS, durante este procedimiento, suceden dos cosas. Primero, se crea una cuenta local en la máquina con el nombre de IUSR_NombreMáquina donde *NombreMáquina* es el nombre del servidor Windows NT. Este *NombreMáquina* lo podemos encontrar en Panel de Control, Red. El tab de Identificación desplegara el nombre del servidor.

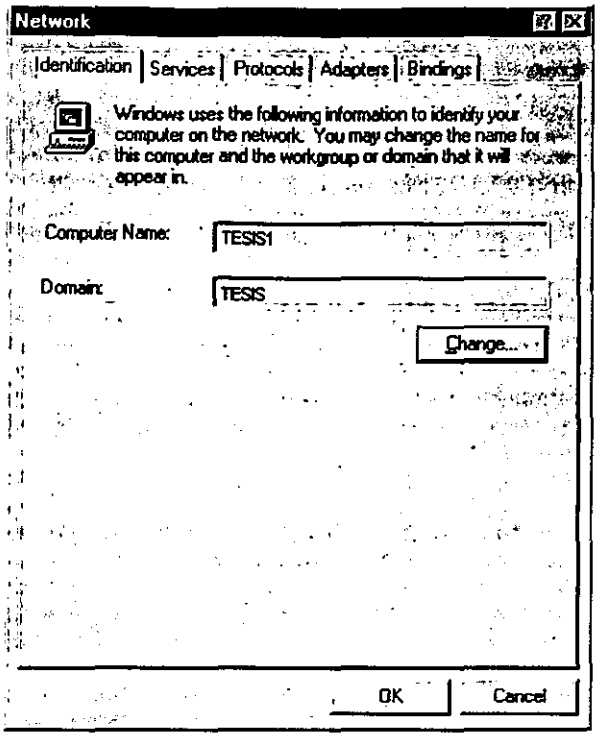


Figura 8.3

En este caso, un Usuario Anónimo con el nombre de IUSR_TESIS1 es dado de alta en la base de datos de usuarios local con una contraseña aleatoria. Esta cuenta es hecha miembro del grupo Invitados (Guests) y le es asignado el derecho de "Log On Locally". Podemos observar esta cuenta haciendo uso del Administrador de Usuarios (User Manager USRMGR:EXE) desde el menú de Inicio, bajo Programas, Herramientas de Administración, User Manager.

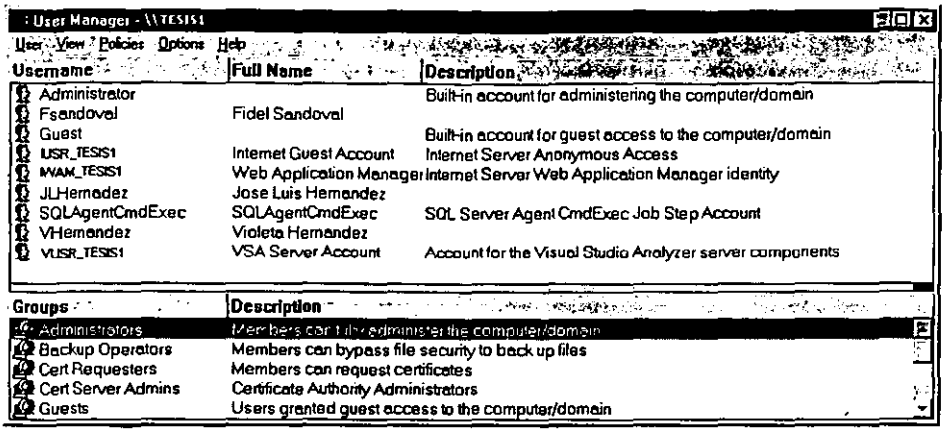


Figura 8.4

Nótese que si IIS es instalado en una máquina que además es Controlador Primario de Dominio (*Primary Domain Controller*) o Controlador Respaldo de Dominio (*Backup Domain Controller*), la cuenta IUSR_NombreMáquina será una cuenta de dominio y será visible en todas las máquinas

Después de crear esta cuenta, IIS se encarga de instalar el servicio de WWW para utilizar esta cuenta. De esta forma instala la cuenta de Usuario Anónimo de forma que cada solicitud por páginas sea hecha bajo el contexto del Usuario Anónimo. Para ver esto es necesario ir a Inicio Programas *Internet Information Server*, y ejecutar el Internet Service Manager.

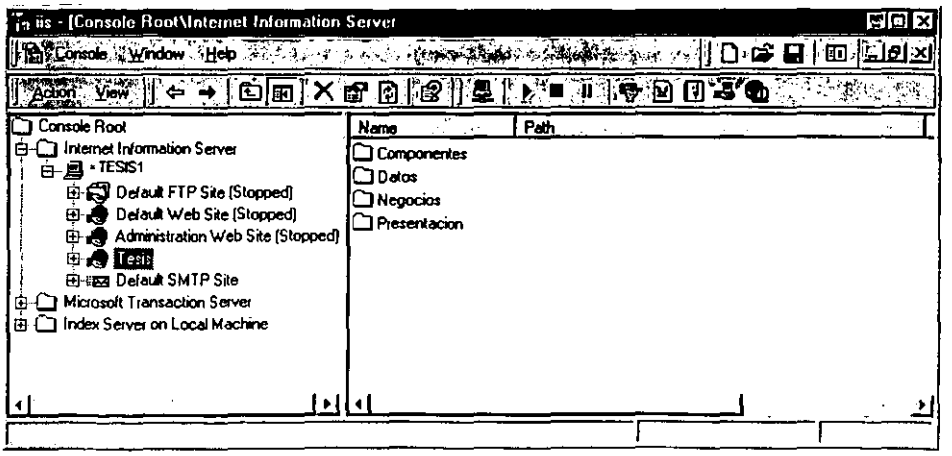


Figura 8.5

Una vez en el Administrador IIS dar doble click en el servicio de WWW sobre el nombre de la computadora.

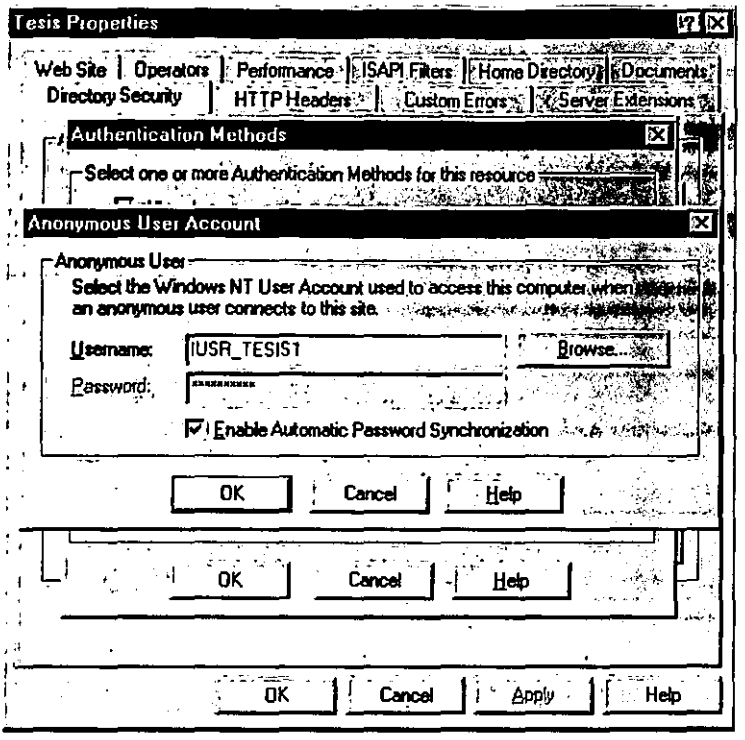


Figura 8.6

La instalación del IIS pone aquí el mismo IUSR_NombreMáquina con la misma contraseña utilizada en el momento de dar de alta la cuenta de usuario local. Con la opción de "Allow Anonymous" habilitada, IIS ejecutará la siguiente secuencia cada vez que una solicitud por páginas sea hecha por un cliente Web:

1. Intenta accesar el archivo solicitado
2. Cuando NT lance el reto, IIS utilizará la información mostrada en la sección "Anonymous Logon" de arriba, como sus credenciales.
3. La máquina NT que contiene el archivo checará estas credenciales contra la base de datos de usuarios local, así como contra la base de datos de usuarios de dominio.
4. Dado que la cuenta IUSR_NombreMáquina fue adicionada como una cuenta local, las credenciales serán válidas.
5. Asumiendo que el archivo es accesible por el grupo Guests o por la cuenta IUSR_NombreMáquina especialmente, entonces se le permite a él IIS acceder al archivo.
6. Se pasa el archivo al cliente Web que lo solicito.

¿QUÉ TAN SEGURO ES EL ACCESO ANÓNIMO?

Un punto importante que se tiene que resaltar aquí es que los privilegios de acceso para cualquier archivo solicitado son siempre verificados aún y cuando se este utilizando la

cuenta de Usuario Anónimo. Esto significa que el uso de la cuenta de Usuario Anónimo compromete en ningún momento la seguridad. Si se desea que un archivo se encuentre a salvo del público en general, lo único que tenemos que hacer es restringir el acceso a ese archivo haciendo uso de los permisos de Windows NT.

El acceso anónimo con frecuencia pone en alerta a los administradores de sistemas, especialmente cuando un componente necesita accesar al directorio System32. Debemos notar, sin embargo, que el usuario autenticado anónimamente no tiene acceso a la contraseña utilizada en la autenticación anónima. Además, tales usuarios tienen acceso solamente a los archivos que han sido explícitamente compartidos por el IIS. Con esto en mente, podemos ver que existen múltiples barreras para prevenir que un extraño tome ventaja del acceso anónimo para accesar recursos restringidos. En tal caso, por principio, no será capaz de accesar a los archivos, porque los archivos no están visibles a él. Aún y cuando pudiese tener acceso a los archivos, no podrá completar la autenticación porque no tiene toda la información necesaria de la cuenta para tener acceso a los archivos que busca. Una vez más esto nos lleva al mismo punto: El uso de la cuenta IUSR_NombreMáquina por ningún motivo compromete la seguridad.

AUTENTICACIÓN NT CHALENGE/RESPONSE

NT Chalenge/Response es el método que el IIS utiliza en el caso de que la cuenta IUSR_NombreMáquina no pueda tener acceso a un archivo en particular. Una vez que se le niega el acceso a la cuenta IUSR_NombreMáquina sobre un archivo, IIS lanza un reto al Web browser pidiendo las credenciales para tener acceso al archivo. Si el browser soporta la autenticación NT Chalenge/Response, entonces proporcionará las credenciales a la máquina IIS. Los beneficios de este método de autenticación son:

- ✓ Las contraseñas no son transmitidas por la red
- ✓ Si el usuario se encuentra trabajando en un máquina Windows, las credenciales son proporcionadas automáticamente sin interacción adicional por parte del usuario.

NT Chalenge/Response requiere de una conexión persistente entre el cliente y el servidor Web. Esto se lleva a cabo a través de sockets TCP/IP. Cuando se solicitan las credenciales la máquina IIS indicará al browser que requiere un Socket abierto. Si el Socket se pierde o se cierra, entonces el proceso de autenticación NT Chalenge/Response fallará. Esto puede ser un problema si un servidor Proxi o Firewall no esta configurado para manejar esta situación. Debido a esto, la autenticación NT Chalenge/Response es útil primordialmente en un ambiente de intranet a diferencia de un ambiente Internet real.

Actualmente sólo el Internet Explorer soporta este método de autenticación.

AUTENTICACIÓN BÁSICA

Si la autenticación NT Chalenge/Response no funciona, o si el Web browser hace uso de este método, la autenticación básica tendrá lugar si es que se encuentra habilitada. Este método ocasiona que el Web browser despliegue una caja de diálogo solicitando el nombre de usuario y la contraseña para pasarlos a la máquina IIS. Esta información es Base64-encoded y transmitida por la red. Las credenciales del usuario son entonces, visibles a cualquiera que pueda estar monitoreando las interacciones entre el Web browser y el Web server.

Después de que las credenciales del usuario han sido enviadas por la red, IIS utiliza estas credenciales localmente para emular ese usuario y accesar los recursos en cuestión. El tener las credenciales localmente puede ayudar en la solución de problemas potenciales, como veremos posteriormente cuando se discuta SQL Server.

AUTENTICACIÓN HTTP

El asegurar un documento Web es un ajuste menor en el proceso normal de HTTP Get/Response. A continuación se describe el proceso por el cual los documentos Web son solicitados por los browsers:

1. El Web browser envía una petición HTTP Get al Web server.
2. El Web server evalúa la petición HTTP Get y localiza el documento en cuestión.
3. El Web server abre el documento y lee la información del encabezado contenida en el documento.
4. El Web server hace uso de esta información para crear un encabezado HTTP Response y enviarlo de regreso al Web browser. El Web browser puede usar esta información para indicar al usuario que el documento se esta descargando y el avance en la descarga si el tamaño del documento fue enviado en el encabezado HTTP.
5. El Web server inmediatamente continua con el envio del documento sin importar cuantos paquetes HTTP sean necesarios para enviar el cuerpo del documento.

El aseguramiento de los archivos altera este proceso en forma mínima. En lugar de responder con un encabezado HTTP Response, el Web server negará el acceso al documento enviando un error "HTTP 401 Unauthorized". Dependiendo del método de autenticación que se este usando (Básica o NT Chalenge/Response) la solicitud para la autenticación cambia de forma poco significativa. Todo esto no lleva a la siguiente secuencia:

1. El Web browser envía una petición HTTP Get al Web server.
2. El Web server evalúa la petición HTTP Get y localiza el documento en cuestión.
3. El Web server intenta abrir el documento y se encuentra que el archivo se encuentra restringido a ciertos usuarios.
4. El Web server manda un HTTP Response de regreso con el error "Access Denied". En este encabezado el Web server especificará cuales medios de autenticación soporta. Si acepta NT Chalenge/Response, entonces enviará un NTLM en el encabezado. Si la autenticación Básica se encuentra habilitada, este método de autenticación también enviará un encabezado HTTP. En este punto depende del browser que método utilizará para autenticarse. Microsoft Internet Explorer utilizará NT Chalenge/Response si esta opción fue indicada en el encabezado HTTP, y Netscape Navigator eligirá la autenticación Básica.
5. Si se usa la autenticación Básica, se desplegará una caja de diálogo solicitando el nombre de usuario y la contraseña. Si se usa NT Chalenge/Response el proceso de autenticación se llevará a cabo invisible al usuario si es posible,

   NOTA:
   Esto significa que si se usa la autenticación NT Chalenge/Response y el cliente utiliza un browser diferente al Internet Explorer, el nombre de usuario y la contraseña serán transmitidos por la red, comprometiendo así la seguridad.

6. El Web server usa la información de la caja de diálogo para intentar accesar al archivo solicitado. Si la información proporcionada tiene los privilegios suficientes el Web server continuará con el envío del documento.

Una vez entendido este proceso, entonces podemos visualizar que existen dos oportunidades para forzar a que la autenticación ocurra cuando los clientes solicitan documentos Web. Podemos, también restringir los documentos a nivel de archivo, o podemos forzar el envío del mensaje HTTP 401 haciendo uso del objeto Response.

ASEGURAMIENTO DE PÁGINAS

El control del acceso a una archivo o página determinados se puede manejar de dos formas. La más común es controlando los permisos a los archivos utilizando la seguridad de Windows NT, pero también se puede hacer de forma programática.

ASEGURAMIENTO DE PÁGINAS USANDO LA SEGURIDAD DE WINDOWS NT

Generalmente, los administradores de Web hacen uso de las ACLs para bloquear el acceso a los archivos. Esto se puede hacer archivo por archivo o bien asignando permisos a todo un directorio y a todos los archivos contenidos dentro de este.

Para cambiar los permisos sobre los archivos, es necesario contar con un drive formateado con NTFS.

1. Utilizando el NT Explorer, localice el archivo o directorio que desea asegurar
2. Sobre el archivo o directorio de click derecho, un menú popup es desplegado.

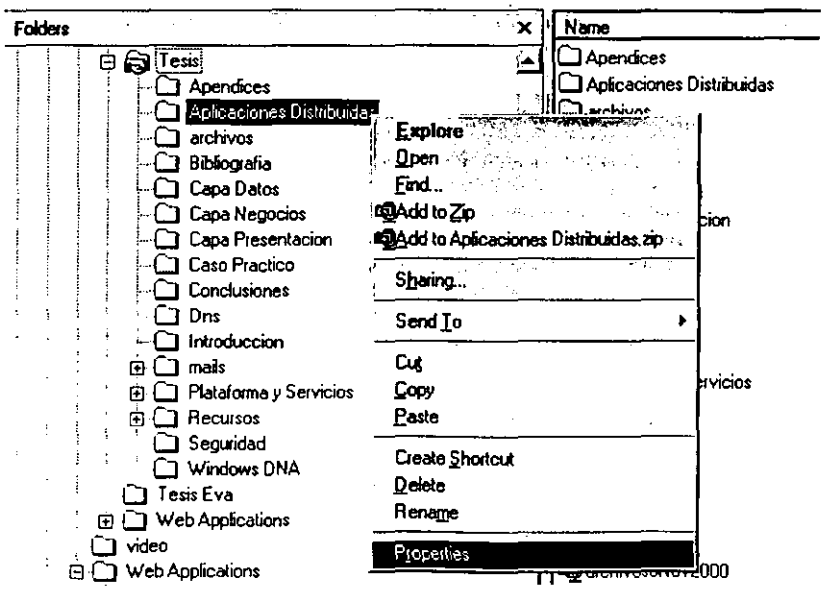


Figura 8.7

3. Seleccione propiedades para desplegar la caja de diálogo de Propiedades que contiene el tab de Seguridad.

4. Seleccione el botón de Permisos. Aparece una lista de todos los usuarios y grupos con privilegios sobre el archivo o el directorio en cuestión.

5. De click en el botón de Agregar para desplegará la caja de diálogo de Adicionar Usuarios y Grupos, es aquí en donde se pueden adicionar usuarios o grupos específicos a la lista de permisos para el archivo o directorio.

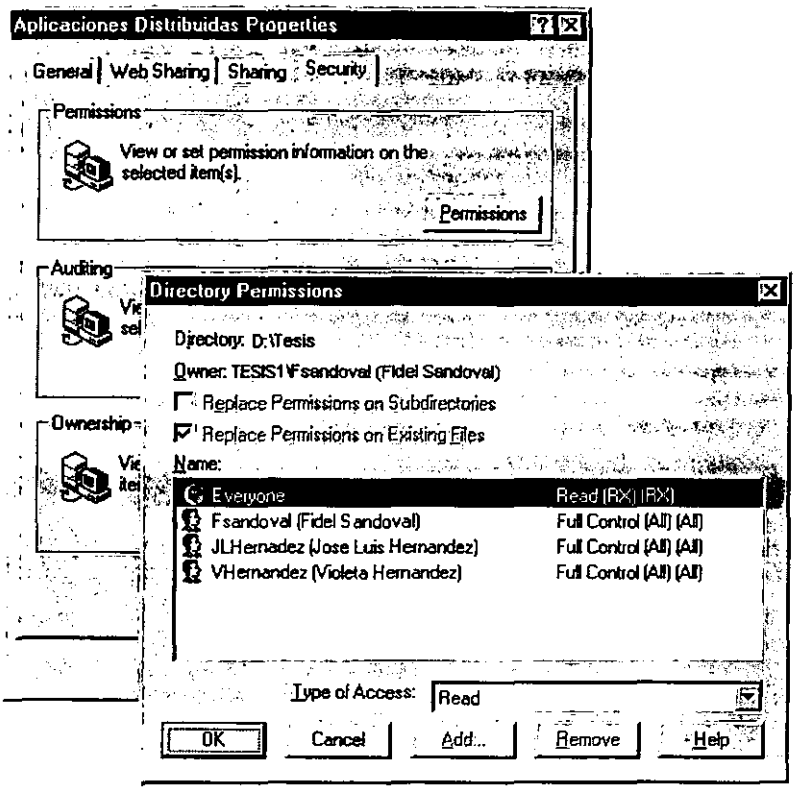


Figura 8.8

ASEGURAMIENTO DE PÁGINAS USANDO HTTP

En ocasiones es necesario controlar de forma programática quien accesará a un archivo en particular. Utilizando el objeto Response de Active Server Pages (ASP) para alterar el encabezado HTTP para negar el acceso a un archivo dado o solicitar las credenciales necesarias. El siguiente script ejemplifica de forma sencilla esta situación:

```
<%
Response.Clear
Response.Buffer = True
```

```
Response.Status = "401 Unauthorized"
Response.AddHeader "WWW-Authenticate","NTLM"
Response.End
%>
```

Este código envía un status HTTP 401 al browser, que reconoce esto como acceso denegado a un archivo dado. Internet Explorer checa que métodos de autenticación son soportados, en este ejemplo se indica que NTLM (NT Chalenge/Response) es soportado. Después de reconocer esto, el Internet Explorer continua con el proceso de autenticación NT Chalenge/Response.

IMPERSONATION

IIS utiliza una característica denominada *Impersonation* cuando despacha las páginas Web. IIS despacha el contenido de las páginas emulando usuarios. Tal es el caso del Usuario Anónimo al utilizar la cuenta IUSR_NombreMáquina. Si se lleva a cabo la autenticación por cualquiera de los dos métodos antes mencionados, IIS emulará al usuario cuyas credenciales fueron proporcionadas durante el proceso de autenticación. Esto significa que cualquier solicitud que se haga a una máquina IIS, en realidad será hecha bajo en contexto de la cuenta IUSR_NombreMáquina, la cuenta bajo la cual se encuentre trabajando, o bien la cuenta cuyo nombre de usuario y contraseña fueron proporcionados durante la autenticación Básica.

Para que tal emulación ocurra, la cuenta a ser utilizada debe tener el derecho de "Log On Locally". Este derecho permite al usuario accesar de forma interactiva a la máquina en cuestión. Este derecho se le otorga a la cuenta IUSR_NombreMáquina cuando es creada durante la instalación. En esencia, cada vez que una página es accesada de forma anónima, la máquina IIS emula la cuenta IUSR_NombreMáquina, la cual accesa a la máquina IIS como si fuese un usuario virtual.

DELEGATION

La autenticación de NT es un intercambio entre el cliente, el servidor encargado de proporcionar el recurso que necesita la autenticación, y el controlador de dominio que valida la acción. Debido a este diseño, los desarrolladores de aplicaciones en IIS deben afrontar esta problemática. Para entender esto, es necesario primero entender los detalles de la autenticación. Podemos describir el proceso de autenticación en la siguiente secuencia de pasos:

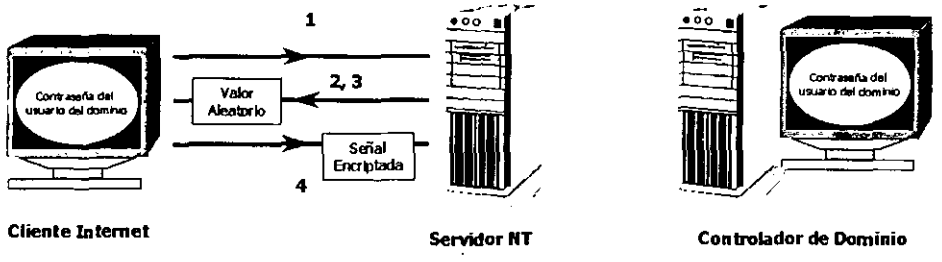


Figura 8.9. Proceso de autenticación, parte uno

1. Se le hace la solicitud de un recurso a un servidor Windows NT.
2. El servidor NT necesita autenticar al usuario antes otorgar acceso al usuario sobre ese recurso.
3. El servidor NT lanza un reto al usuario que solicita ese recurso. En este reto el servidor envía un valor aleatorio, mismo que es encriptado
4. La máquina cliente envía el nombre del usuario, el nombre del dominio y el token encriptado al servidor NT. El token es encriptado con la combinación de un "hash" generado a partir de la contraseña del usuario y el valor aleatorio enviado por el servidor.

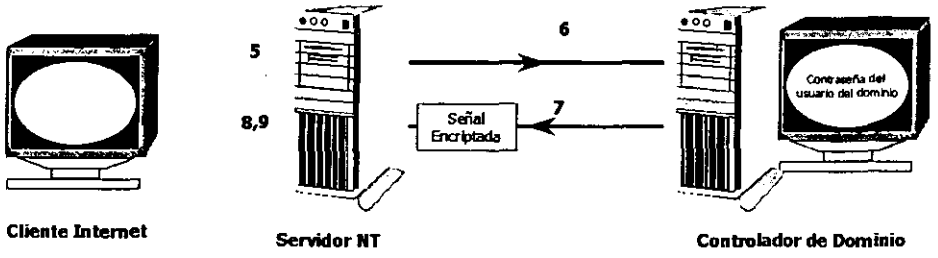


Figura 8.10 Proceso de autenticación, parte dos

5. El servidor NT recibe el token encriptado y lo almacena para uso posterior.
6. El servidor NT utiliza el nombre del usuario, el nombre del dominio proporcionado con anterioridad por la máquina usuario y el número aleatorio generado por el mismo, para solicitar un token al controlador de dominio.
7. El controlador de dominio utiliza esta copia del "hash" de la contraseña del usuario y el número aleatorio como llave, y encripta un token que contiene el nombre del usuario y el nombre del dominio, y lo envía de regreso al servidor NT.
8. El servidor NT compara los token recibidos del controlador de dominio y la máquina cliente para asegurarse de que estos coincidan
9. Si los token coinciden, se le otorga acceso al usuario sobre el recurso en cuestión.

El punto más importante a resaltar en esta interacción es el hecho de que el servidor NT en ningún momento conoce la contraseña del usuario ni el *"hash"* generado a partir de ésta. Dado que el servidor NT no transmite las contraseñas sobre la red, éste se ve obligado a solicitar un token al controlador de dominio. Una vez hecho esto no puede enviarlo a ninguna otra máquina. En el caso de que pudiera enviar este token a otra máquina, esto se conocería como "delegation". Windows NT no soporta esto en la versión 4.0, lo cual ocasiona algunas complicaciones a los desarrolladores Web.

Considérese lo siguiente:

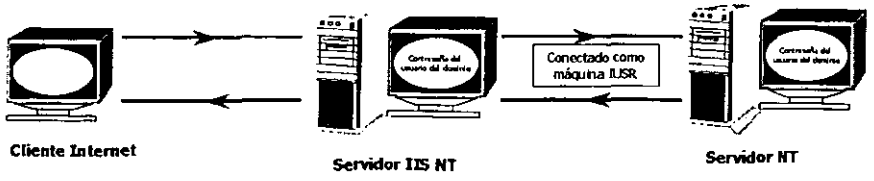


Figura 8.11 Acceso a un recurso como el usuario anónimo.

En un servidor NT bajo circunstancias normales, el cliente Web será autenticado como el Usuario Anónimo (IUSR_NombreMáquina) por la máquina IIS. Recordemos que la cuenta IUSR_NombreMáquina se encuentra en la máquina en la cual esta instalado IIS, así que la autenticación tiene lugar entre la máquina IIS y el controlador de dominio. Ahora consideremos que una página ASP no tiene privilegios para ser vista por IUSR_NombreMáquina. Bajo este escenario el cliente Web debe ser autenticado. Recordemos que cuando IIS procesa una página ASP lo hace bajo el contexto de un usuario autenticado.

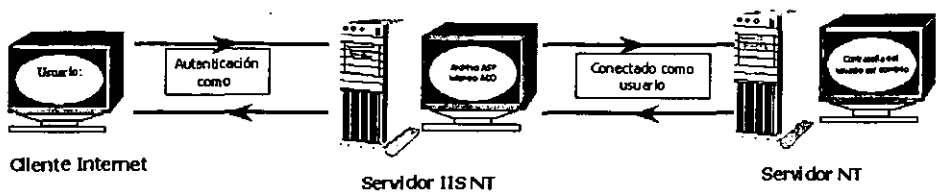


Figura 8.12 Acceso a un recurso como un usuario autenticado.

En este caso, esto significa que la conexión al archivo MDB será hecha bajo el contexto del usuario. Tómese en cuenta que la contraseña de PaulEn no existe en la máquina IIS. Ahora, cuando IIS intenta accesar al archivo MDB, el servidor NT en el cual reside dicho archivo, lanzará un reto al IIS preguntando quien desea accesar el archivo MDB. En este punto, Windows NT comienza el proceso de autenticación descrito anteriormente. El problema aquí ocurre en el paso descrito antes, cuando la máquina IIS debe encriptar el nombre del usuario y el nombre del dominio. Debido a que la máquina IIS no posee la contraseña, no puede llevar a cabo esta tarea. Además debido a que Windows NT 4.0 no soporta "delegation" el token utilizado para autenticar al usuario no puede ser enviado al servidor NT en el cual reside el archivo MDB.

Esta limitante la podemos resumir en: "Los recursos que requieren de autenticación para poder ser accesados no podrán accesar recursos localizados físicamente en otra máquina al menos que la máquina IIS sea un controlador de dominio".

Nótese que utilizamos la palabra "recurso" y que la explicación previa fue utilizando un script ASP, pero es precisamente porque podemos extender esto a escenarios diferentes como podría ser el caso de los componentes Activex.

También nótese que este inconveniente no aplica si la máquina IIS es un controlador de dominio. Esto es porque como controlador de dominio tiene acceso local a las cuentas y tendrá la capacidad de completar apropiadamente el proceso de autenticación.

COMPONENTES COM

Las *Active Server Pages* (ASP) tienen la capacidad de instanciar objetos Activex (Componentes COM/DCOM) desde un script. Normalmente los desarrolladores construyen controles Activex para usarlos como objetos de negocios para accesar recursos en otras máquinas. Cuando son usados en una página que requiere autenticación para ser visualizada, podemos caer en el mismo problema si no se tiene el debido cuidado en la creación de estos componentes.

Como sabemos ahora, los recursos son accesados bajo el contexto de un usuario emulado. Esto también se extiende a los componentes que son instanciados desde un script en una ASP. La capacidad de estos componentes para tener acceso a otros recursos depende del contexto del usuario bajo el cual hayan sido instanciados, lo cual puede ser directamente afectado por el modelo de *"threading"* con el cual dicho componente fue creado.

COMPONENTES DCOM

Los componentes DCOM difieren en mínimamente de las aplicaciones en la forma en cómo estos son usados. Debido a que estos componentes pueden ser instanciados sobre la red o bien en máquina remotas, niveles de seguridad extra se introducen. Estos niveles de seguridad son en la forma de permisos de seguridad definidos en las llaves del Registry necesarias para instanciar el componente. Debido a la complejidad de esto, se nos proporciona la herramienta DCOMCNFG.EXE para facilitar la configuración de estos permisos.

En la implementación de los componentes DCOM, es importante no perder la perspectiva del contexto del usuario bajo el cual el componente será creado. Con esto en mente, podemos asignar apropiadamente los permisos utilizando DCOMCNFG.EXE de tal forma que el usuario pueda instanciar tantos objetos DCOM como desee.

THREADING

Los componentes desarrollados para el IIS deben ser sensitivos al modelo de *"threading"* bajo el cual fueron desarrollados. Los componentes son instanciados bajo el contexto de un usuario determinado por el modelo de *"threading"*. Es decir, los componentes *"single-thread"* correrán bajo el contexto de la cuenta de sistema local, los *"apartment-model"* y los *"free-thread"* marcados en el Registry como *"Both"* correrán bajo el contexto del usuario emulado por el IIS.

Para determinar esto, IIS verifica en el Registry del sistema para obtener el modelo de *"threading"* del componente. Un componente puede ser marcado con una llave *"ThreadingModel"*, cuyos valores válidos son: *"Apartment"* y *"Both"*. Un ejemplo de esta configuración es el siguiente:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{0000022C-0000-0010-8000-
00AA006D2EA4}\InprocServer32] "ThreadingModel"="Both"
```

COMPONENTES EN VISUAL BASIC 6.0

Visual Basic tiene la habilidad de crear componentes en el modelo *"Apartment"*. Para hacer esto, se debe configurar el proyecto de forma especial. Para un componente Activex DLL dado, debemos hacer lo siguiente:

1. Abrir el proyecto vbp.
2. Bajo el menú de Project, seleccionar Propiedades.
3. En el tab de General, marcar el checkbox de Unattended Execution.

4. Después de compilar el componente, para convertirlo al modelo de "*threading*" de "*Apartment*" necesitamos adicionar la llave "*ThreadingModel*" en el Registry del sistema como se mostró en el ejemplo anterior.

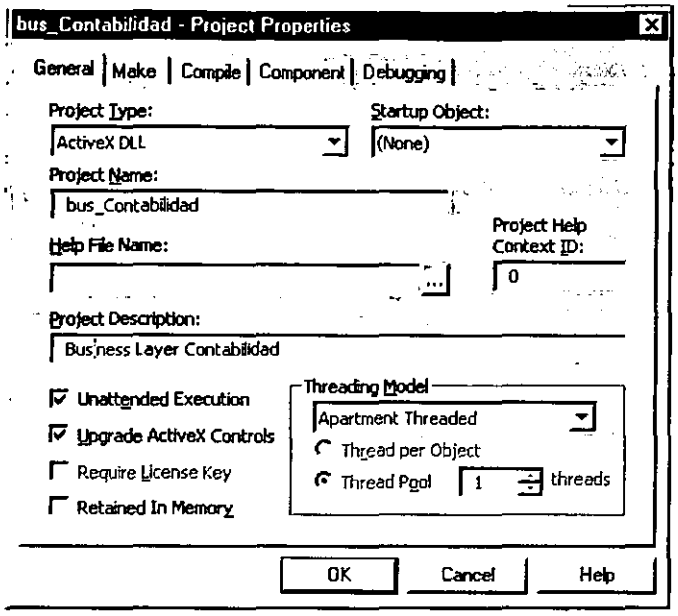


Figura 8.13

COMUNICACIONES SEGURAS USANDO SECURED SOCKET LAYER (SSL)

Una vez que se ha establecido control sobre quien puede y quien no puede ver nuestras páginas Web, necesitamos asegurar que cualquier información enviada a o recibida de un sitio Web permanezca confidencial. Bajo un canal de comunicación HTTP normal, cualquier individuo con la capacidad de observar él trafico en Internet tendrá la oportunidad de observar la información que enviamos. Para solucionar este problema, Netscape introdujo *Secured Socket Layer* (SSL). La tecnología equivalente de Microsoft es *Private Comunication Technology* (PCT) y es compatible con SSL debido a que ambas comparten el formato X.509. Mientras que esta es una tecnología diferente, la mayoría de la gente se refiere, por simplicidad, a los canales seguros como SSL.

Active Server Pages proporciona herramientas que ayudan a la integración del sitio Web con SSL. SSL se encargará de los detalles de definir la identidad de un cliente Web, mientras que ASP permite determinar que tipo de tareas le son permitidas una vez que se ha definido. Por lo tanto, una vez que SSL se encuentra instalado y trabajando apropiadamente, todas las comunicaciones entre el sitio Web y sus clientes serán encriptadas y completamente confidenciales. Esto por su puesto, asume que cliente se encuentra conectado invariablemente vía SSL, para cual se requiere el uso del identificador HTTPS en el URL, como se muestra a continuación.

```
HTTPS://Tesis1/Presentacion/Polizas/ContaPoliza.asp
```

SSL tiene una repercusión en el rendimiento al despachar el contenido Web, debido al trabajo de encripción adicional que se tiene que llevar a cabo antes de enviar la información. La información que no necesita ser transmitida sobre un canal seguro, debe manejarse por la vía HTTP normal. Un ejemplo de esto puede ser cualquier archivo de imagen (GIFs, JPEGs) utilizado en el sitio Web.

El acceso de la información desde un cliente Web se lleva a cabo a través del objeto Request, el cual es un objeto ASP intrínseco. El método Request.ClientCertificate permite examinar la información del cliente Web y determinar si se le permitirá o no el acceso a las páginas. Esto proporciona un método alternativo de autenticación de un usuario, independiente de la seguridad normal de NT.

SEGURIDAD EN SQL SERVER

SQL Server proporciona otro nivel de complejidad en la implementación de la seguridad con IIS. Dependiendo del nivel de integración con IIS requerido, se necesitará configurar a SQL para alcanzar tal nivel. Las limitantes de "*delegation*" de Windows NT 4.0 afectan directamente aquí, así que es necesario entender como es que la seguridad de SQL Server se relaciona con la seguridad de Windows NT.

SEGURIDAD EN SQL SERVER VS SEGURIDAD EN WINDOWS NT

SQL Server soporta tres modelos diferentes de seguridad. Estos modelos permiten completar la integración con el modelo de dominios de NT, no integración con el modelo de dominios de NT, o una mezcla de ambos.

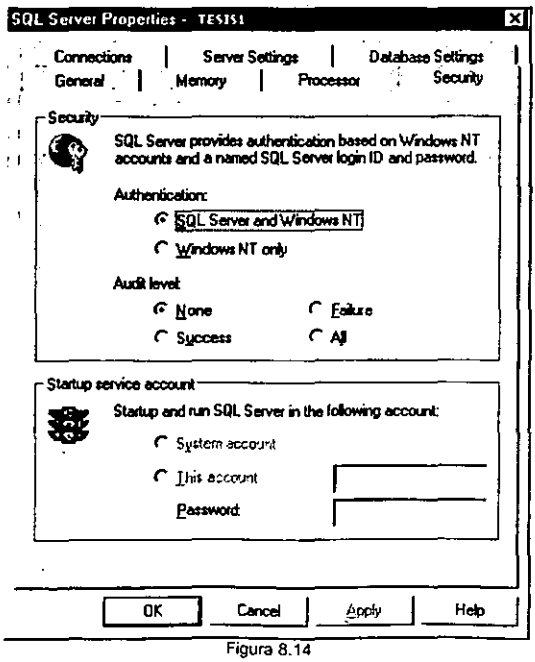


Figura 8.14

SEGURIDAD STANDARD

La seguridad standard es la opción de default al instalar SQL Server. Este modelo de seguridad es el más sencillo porque la seguridad es completamente independiente del modelo de dominios de NT. El nivel de acceso que tiene un usuario a la base de datos y sus objetos se encuentra determinado por las configuraciones de seguridad dentro del mismo SQL Server. La autenticación consiste en la comparación del nombre de usuario y la contraseña contra información similar que se encuentra residente en la base de datos de SQL Server. Este es el modelo de seguridad más fácil de integrar con IIS.

SEGURIDAD INTEGRADA

La seguridad integrada se encuentra en el polo opuesto a la seguridad standard. Este modelo de seguridad es completamente dependiente de la seguridad de Windows NT. Por ejemplo cuando un usuario intenta conectarse a SQL Server, la conexión será hecha bajo el nombre de usuario que se encuentre trabajando en la máquina en ese momento. La seguridad integrada requiere de que SQL Server sea configurado para usar Named Pipes como protocolo de red para la comunicación. El proceso de autenticación se realiza de la misma manera como hemos descrito con anterioridad en la sección de "Delegation". La integración con IIS puede presentar problemas, debido a la ausencia de "delegation" en Windows NT 4.0

SEGURIDAD MIXTA

La seguridad mixta ofrece una combinación de la seguridad standard y la seguridad Integrada. La forma en la que un usuario intente establecer una conexión con SQL Server determinará el método de autenticación a usar. Básicamente, si el usuario se conecta vía ODBC, la autenticación será vía el proceso de autenticación de Windows NT. Si el usuario no se conecta vía ODBC, la seguridad standard será utilizada, y el nombre de usuario y la contraseña proporcionados, simplemente serán cotejados contra la base de datos de usuarios de SQL Server.

PROTOCOLOS DE SQL SERVER

Dependiendo del protocolo utilizado por SQL Server se tendrá un gran impacto en la integración de SQL Server con IIS. No obstante SQL Server ofrece una gran variedad de protocolos, Named Pipes y TCP/IP sockets son los más usados.

NAMED PIPES

Como se mencionó anteriormente, *Named Pipes* es un protocolo autenticado. Esto significa que siempre que un usuario intenta abrir una conexión con SQL Server vía *Named Pipes*, el proceso de autenticación de NT tendrá lugar. Es importante recordar esto, debido a que *Named Pipes* es el protocolo de default cuando se instala SQL Server. Además se requiere de *Named Pipes* para configurar a SQL Server con seguridad integrada.

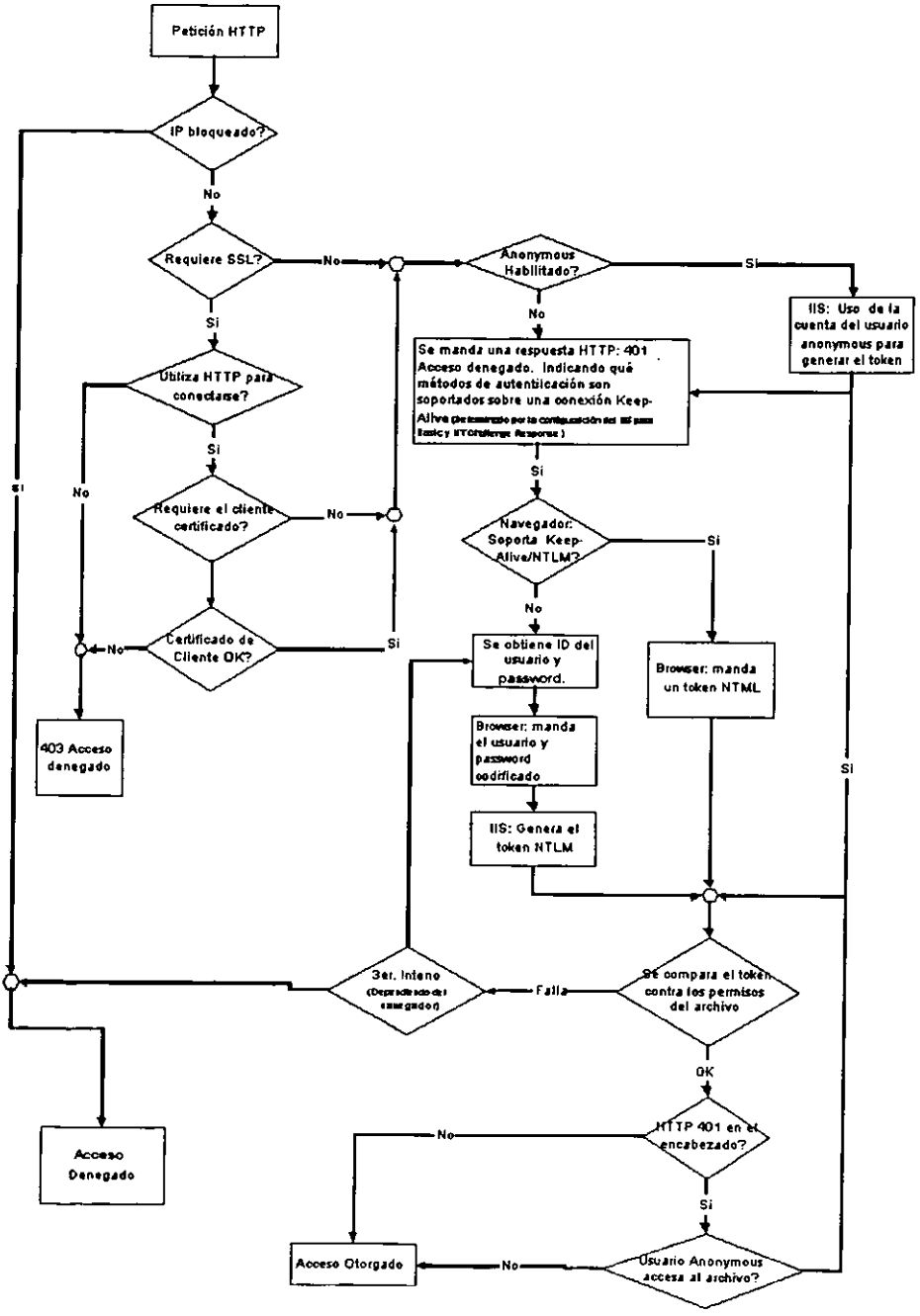

Petición HTTP
IP bloqueado?
No
Si
Requiere SSL?
No
Si
Anonymous Habilitado?
Si
No
IIS: Uso de la cuenta del usuario anonymous para generar el token
Se manda una respuesta HTTP: 401 Acceso denegado. Indicando qué métodos de autenticación son soportados sobre una conexión Keep-Alive
Si
Utiliza HTTP para conectarse?
Si
No
Si
Requiere el cliente certificado?
No
Navegador: Soporta Keep-Alive/NTLM?
Si
No
No
Certificado de Cliente OK?
Si
Se obtiene ID del usuario y password.
Browser: manda un token NTML
403 Acceso denegado
Browser: manda el usuario y password codificado
IIS: Genera el token NTLM
Si
Ser. Inteno
Falla
Se compara el token contra los permisos del archivo
OK
Acceso Denegado
No
HTTP 401 en el encabezado?
Si
Acceso Otorgado
No
Usuario Anonymous accesa al archivo?

Figura 8.15

TCP/IP SOCKETS

Sockets es un protocolo no autenticado. Esto significa que las conexiones no serán autenticadas y que las conexiones y los niveles de acceso son determinados solamente por el nombre de usuario y la contraseña proporcionados durante la conexión.

USANDO SQL SERVER CON IIS

La integración de SQL Server con IIS es la culminación de todas ésta cuestiones de seguridad, contenidas en un solo tema. Todos los factores antes descritos, vienen a tomar parte aquí, incluyendo que configuración de seguridad este utilizando IIS, como se encuentre configurado SQL Server, y la localización física de cada herramienta. Muchos de estos retos de integración provienen de la incapacidad que Windows NT 4.0 presenta para la "*delegation*". Con la introducción de Kerberos en Windows 2000, muchos de estos inconvenientes han quedado en el pasado. Por ahora, revisaremos algunas configuraciones comunes y veremos como podemos hacer funcionar las cosas y afrontar estos retos.

SEGURIDAD STANDARD DE SQL SERVER

Como se mencionó anteriormente, la seguridad standard representa un número mínimo de retos cuando se hace la integración con IIS. Una parte importante a considerar es si se usa o no un protocolo autenticado.

Siendo *Named Pipes* un protocolo autenticado, forza a que ocurra una autenticación cada vez que intenta establecer una conexión. Debido a que esto pasa desapercibido a nosotros es un punto difícil de localizar cuando nos encontramos configurando o resolviendo problemas de conectividad de SQL Server. Para visualizar el impacto de esto, veamos la secuencia de eventos que ocurren cuando se hace una solicitud de una página que hace una conexión a SQL Server.

SEGURIDAD STANDARD CON SQL SERVER Y IIS EN LA MISMA MÁQUINA

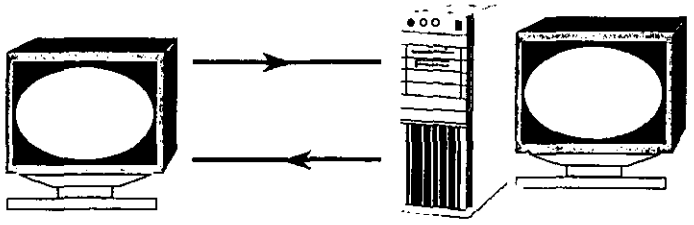


Figura 8.16

1. El cliente solicita una página ASP que hacen acceso a datos
2. IIS hace la solicitud del archivo a NT, el cual inicia el proceso de autenticación
3. El usuario es autenticado como IUSR_NombreMáquina, si la autenticación anónima se encuentra habilitada, o bien es autenticado como el usuario cuyo nombre de usuario y

contraseña fueron proporcionados si la autenticación anónima se encuentra deshabilitada y la página requiere de autenticación.

4. IIS intenta establecer una conexión desde el archivo ASP con SQL Server vía *Named Pipes*
5. La solicitud de conexión es recibida por SQL Server (en la misma máquina), quien a su vez solicita la autenticación debido a que *Named Pipes* es un protocolo autenticado.
6. El proceso de autenticación se lleva a cabo todo en la misma máquina. Nótese que cuando SQL Server intenta validar la cuenta IUSR_NombreMáquina, esta se encuentra localizada en la base de datos de usuarios local y en consecuencia no existe ningún problema. De forma similar , si IIS se vio en necesidad de autenticar al usuario para darle acceso a la página ASP (o si la autenticación anónima estaba deshabilitada), el token para esta autenticación se encuentra en la misma máquina en la que se encuentra instalado SQL Server, de manera que nuevamente no existe ningún problema.
7. Después de que la conexión con *Named Pipes* es establecida, el nombre de usuario y la contraseña son usados para determinar el nivel de acceso a los recursos de SQL Server.

En resumen, al encontrarse SQL Server instalado en la misma máquina que IIS, físicamente en el mismo servidor NT no existen problemas de integración entre estos productos.

SEGURIDAD STANDARD CON SQL SERVER Y IIS EN MÁQUINAS DIFERENTES

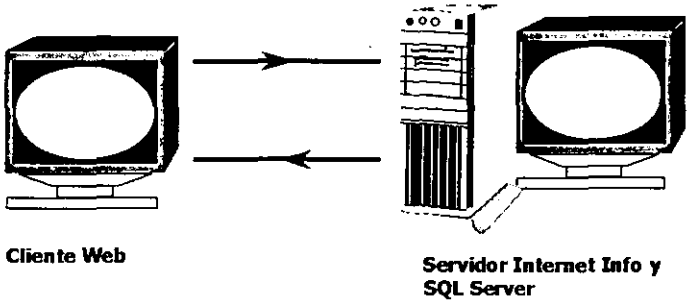


Figura 8.17

1. El cliente solicita una página ASP que hacen acceso a datos.
2. IIS hace la solicitud del archivo a NT, el cual inicia el proceso de autenticación
3. El usuario es autenticado como IUSR_NombreMáquina, si la autenticación anónima se encuentra habilitada, o bien es autenticado como el usuario cuyo nombre de usuario y contraseña fueron proporcionados si la autenticación anónima se encuentra deshabilitada y la página requiere de autenticación.
4. IIS intenta establecer una conexión con SQL Server emulando a IUSR_NombreMáquina o a un usuario autenticado.
5. SQL Server lanza un reto a la máquina IIS iniciando el proceso de autenticación, debido a *Named Pipes*
6. Si IIS procesa la página bajo la cuenta de IUSR_NombreMáquina, esta conexión no ocasionará problemas. Esto es debido a que IIS tiene toda la información de la cuenta

IUSR_NombreMáquina localmente, y puede proporcionar la información necesaria para la autenticación. Sin embrago, si IIS intenta establecer la conexión como un usuario autenticado, en este caso no cuenta con una copia local de la contraseña (al menos que este sea un Controlador Primario de Dominio), y en consecuencia no puede crear el token encriptado necesario para establecer la conexión. Por lo tanto, no es posible establecer una conexión con SQL Server vía *Named Pipes* si un archivo ASP requiere de autenticación para poder ser accesado, al menos que IIS se encuentre instalado en un PDC, la conexión será establecida sobre local Pipes, o bien que IIS utilice la autenticación Básica en lugar de NT Chalenge/Response.

7. Después de que IIS pasa el token a SQL Server para la autenticación, SQL Server necesitará verificar dicho token. Para esto seguirá la siguiente secuencia.
   - ✓ Checar la base de datos de usuarios local para obtener la información de la cuenta
   - ✓ Preguntar al controlador de dominio
   - ✓ Checar si la cuenta de Guest se encuentra habilitada
8. Si SQL Server puede verificar el token obtenido de IIS por cualquiera de los pasos mencionados, la conexión vía *Named Pipes* es establecida.
9. Después de que la conexión vía *Named Pipes* es abierta, SQL Server valida el nombre de usuario y la contraseña obtenidos contra la base de datos de usuarios interna. Esta validación es el paso final necesario para accesar los recursos de SQL Server.

Es importante mencionar que el lugar donde SQL Server busca la información de la cuenta de usuario cuando intenta validar la conexión *Named Pipes*, es en el paso 7. Bajo circunstancias normales, la validación fallaría en el paso 7, esto es porque por default IIS utiliza IUSR_NombreMáquina, la cual es una cuenta de usuario local. Esto significa que cuando SQL Server intenta validar la conexión vía *Named Pipes*, no será capaz de encontrar la información de la cuenta y por lo tanto no será capaz de establecer la conexión.

USO DE OTROS PROTOCOLOS

Otros protocolos facilitan los inconvenientes en la implementación entre IIS y SQL Server de forma considerable. Con la ausencia de un protocolo autenticado, el acceso es determinado enteramente por el nombre de usuario y la contraseña proporcionados durante la conexión. En este caso no se tiene el problema de "*delegation*".

USANDO SEGURIDAD INTEGRADA EN SQL SERVER

Al utilizar SQL Server con seguridad integrada, nos encontramos nuevamente con un gran reto, esto es debido a que Windows NT 4.0 no soporta "*delegation*". Existen algunos escenarios específicos en los que se puede utilizar seguridad integrada, lo importante es tener en mente que cuando se considere la implementación de la seguridad integrada, la autenticación se apoya directamente en el protocolo *Named Pipes*.

SEGURIDAD INTEGRADA CON SQL SERVER Y IIS EN LA MISMA MÁQUINA

Este escenario de seguridad integrada es en esencia idéntico al descrito anteriormente donde se consideró la seguridad standard con SQL Server y IIS en la misma máquina. La diferencia aquí es que dado que ambos esquemas se llevan a cabo en la misma máquina, no existe la necesidad de pasar el token hacia otra máquina. Debido a esto, SQL Server tiene la posibilidad de autenticar la conexión y por lo tanto no tenemos ningún problema.

Las páginas que hace acceso a datos, no deben permitir el acceso anónimo. Restringiendo el acceso a estas páginas, el proceso de autenticación tendrá lugar cuando el usuario intente abrir dichas páginas. Esto ocasionará que la página ASP sea procesada bajo el contexto de un usuario autenticado y en consecuencia la conexión a SQL Server será hecha bajo el contexto de este usuario.

NOTA:
Esta configuración requiere de una conexión *"pipe"* local para poder funcionar. Si no se usa esta conexión, IIS y SQL Server se comportarán como si se encontrarán en máquinas separadas.

Si se requiere de permitir el acceso del usuario anónimo a las páginas que hacen acceso a datos, se deberá crear una cuenta de usuario anónimo en SQL Server y mapear la cuenta de IUSR_NombreMáquina a esta cuenta.

Nota: Por default SQL Server no permite guiones bajos (_) en los nombres de usuarios, de tal forma que será necesario crear una nueva cuenta de usuario anónimo para el uso de la seguridad integrada con SQL Server. También nótese que esto impide de forma sutil en primera instancia el uso de la seguridad integrada con SQL Server. Si se tiene a todos los usuarios autenticados como anónimos, entonces es mejor implementar la seguridad standard.

SEGURIDAD INTEGRADA CON SQL SERVER Y IIS EN MÁQUINAS DIFERENTES

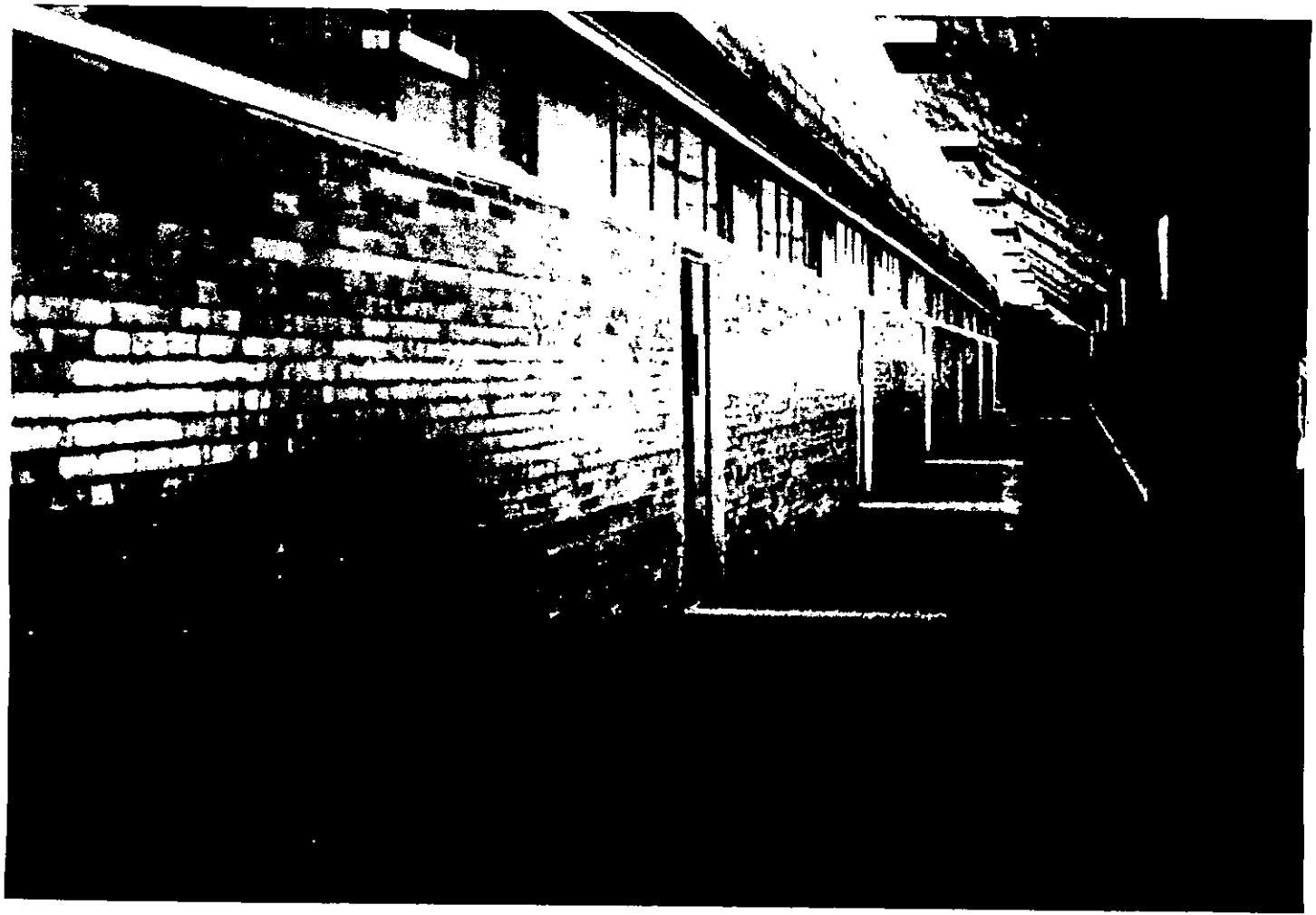
Como recordaremos cuando tratamos el tema de *"Delegation"*, establecimos que: Los recursos que requieran de autenticación, no podrán ser accesado si se encuentran físicamente en otra máquina NT al menos que la máquina IIS sea un controlador de dominio.

Cuando trasladamos esto a un esquema de seguridad integrada, esto implica que las páginas ASP's que no permiten el acceso anónimo no podrán conectarse a SQL Server que se encuentre físicamente en otra máquina NT, al menos que la máquina IIS sea un controlador de dominio, o que se use la autenticación básica, en lugar de Chalenge/Response.

En otras palabras, no es posible tener acceso a páginas Web y conectarse a SQL Server utilizando cuentas de usuarios. Esta limitante es debido una vez más a la cuestión de que Windows NT 4.0 no soporta *"Delegation"*. Cuando IIS intente hacer una conexión Named Pipes con SQL Server, no podrá encriptar un token debido a que IIS no tiene la contraseña de usuario necesaria como llave de encripción (al menos de que este sea un controlador de dominio).

Como hemos podido notar existen un gran número de variables involucradas en el aseguramiento de un sistema. Todas esta variables tienen una aplicación específica de acuerdo a la configuración de hardware y software con que se cuente, así también influye en alcance del sistema y con cuáles tecnologías se este desarrollado. Una vez definido la estructura, el hardware, el software y que tecnologías son idóneas para el desarrollo de un sistema, entonces se puede definir un esquema de seguridad, que se debe implementar desde el inicio del desarrollo del sistema hasta su implementación sobre uno o más servidores.

Para nuestro caso práctico más adelante veremos que el conjunto de hardware y software, así como las tecnologías que decidimos utilizar para el desarrollo.



CAPÍTULO NUEVE

CASO PRÁCTICO

El propósito de la presentación de este caso práctico es ilustrar el proceso en el diseño y desarrollo de un DNS construido bajo la especificación de Windows DNA utilizando las metodologías, técnicas y teoría presentadas durante el desarrollo de esta tesis.

Hemos decidido tomar como modelo a una organización para la implementación del caso práctico y es conveniente dar un breve preámbulo sobre la misma.

PROBLEMÁTICA

La organización es un consorcio de empresas dedicadas a la promoción y desarrollo de vivienda, por lo que uno de sus giros es la construcción de casas y departamentos. Esta empresa exige de una gran eficacia en el manejo de sus operaciones, entre las que destacan el manejo del inventario en los almacenes, el avance de las obras en construcción, el seguimiento de clientes, la cobranza, manejo de los presupuestos para las diferentes obras en proceso, la aplicación contable de todas las operaciones en las empresas y entre las mismas, presentación de estados financieros a la BMV, etc.

El consorcio de empresas cuenta con desarrollos en diferentes puntos del país y tiene la necesidad de controlar todo su flujo de información desde sus oficinas centrales. Capturar todas las operaciones en los diferentes fraccionamientos y enviarlas en tiempo real a oficinas centrales es una necesidad. Los altos directivos de la organización se encuentran siempre visitando los diferentes desarrollos y tienen la necesidad de hacer análisis comparativos con otros desarrollos del mismo consorcio, para lo cual se ayudan de empleados altamente capacitados en el funcionamiento de la empresa. Dichos empleados requieren del acceso a la información de otros desarrollos que se encuentran distanciados geográficamente. Es importante mencionar que el manejo de bases de datos distribuidas en los diferentes fraccionamientos y el replicarlas a oficinas centrales es una opción que está descartada debido a los costos que esto involucra. La infraestructura de hardware con que cuenta la empresa son PC's, servidores y una línea de acceso a Internet dedicada. La opción es una aplicación Windows DNA.

Enfocaremos el desarrollo del caso práctico a una parte esencial en la construcción de un DNS. Dicho subsistema se encarga del manejo contable de la organización, y es el centro del DNS, ya que es aquí donde se ven reflejadas todas las operaciones de la organización en forma de cargos y abonos, entradas y salidas la parte contable es la encargada de administrar y analizar todo el capital, activos y pasivos, de la organización para actuar como un instrumento que le permita a la organización actuar anticipada y oportunamente a los cambios o imprevistos que se puedan suscitar. Y más aún que le permita encontrar nuevas oportunidades para el crecimiento de la organización.

Debido a la magnitud de un DNS, no es posible presentar el desarrollo de un DNS en su totalidad, pero creemos que el presentar el desarrollo de una parte del componente contable es un muy buen ejemplo, y hemos escogido las partes que más ilustren y pongan en práctica las técnicas y teorías presentadas en el desarrollo de esta tesis.

A continuación se presentan la serie de pasos que servirán para el desarrollo del caso práctico:

- ✓ Paso 1. Configuración de los sistemas de hardware y software requeridos
- ✓ Paso 2. Creación del modelo de datos (Capa de Datos)
- ✓ Paso 3. Definición de las reglas de negocios (Capa de Negocios)
- ✓ Paso 4. Creación de los objetos ejecutantes (Executants)
- ✓ Paso 5. Creación de los objetos emisarios (Emissaries)
- ✓ Paso 6. Creación de la interfaz de usuario (Capa de Presentación)
- ✓ Paso 7. Implementación y configuración del sistema

PASO 1. CONFIGURACIÓN DE LOS SISTEMAS DE HARDWARE Y SOFTWARE REQUERIDOS

En el capítulo cuatro[1], se hizo referencia a los componentes esenciales para la implementación de una aplicación Windows DNA. Una vez mencionados estos componentes, resulta necesario definir el orden de instalación del hardware y software.

La configuración del hardware puede variar de acuerdo a las necesidades y la cantidad de información que se maneja dentro de una empresa. Una configuración típica con resultados comprobables dentro de una empresa mediana es la utilización de servidores con las siguientes especificaciones:

Tabla 9.1 Configuración básica de hardware para los servidores

CARACTERÍSTICA	CAPACIDAD
Procesadores	2
Velocidad	500 Mhz
Capacidad de almacenamiento	18 GB
RAM	500 MB

Vale la pena mencionar que esta configuración no es la más poderosa en el mercado actualmente y que por lo tanto se tiene posibilidad de escalar.

Durante el desarrollo de la tesis se definió que las aplicaciones Windows DNA son aplicaciones distribuidas y que esta distribución es tanto física como lógica, por tal razón, la recomendación es tener dos servidores dedicados. Uno se encargará del manejo de la lógica de negocios y el otro tendrá la responsabilidad del manejo de los datos. La configuración en software de cada uno de los servidores se presenta a continuación:

Tabla 9.2 Configuración básica de software para los servidores

SERVIDOR DE NEGOCIOS	SERVIDOR DE BASE DE DATOS
Windows NT	Windows NT
Internet Information Server	Data Access Components (MDAC)
Transaction Server	SQL Server
Data Access Components (MDAC)	

[1] La Plataforma y los Servicios

PASO 2. CREACIÓN DEL MODELO DE DATOS (CAPA DE DATOS)

Un modelo de datos es una descripción detallada de las diferentes entidades de datos requeridas por una aplicación. El proceso de modelado de datos es la identificación, documentación y la implementación que normalmente involucra:

- ✓ Identificar cada entidad de datos requerida para llevar a cabo las operaciones de negocio que pretende automatizar la aplicación
- ✓ Definir el tipo de dato, el tamaño y los valores de default de cada campo
- ✓ Definir las relaciones entre las diferentes entidades (Por ejemplo póliza contable debe contener una o más cuentas contables)
- ✓ Definir la integridad de datos para cada identidad (Por ejemplo el periodo contable se compone de 5 dígitos exactamente, la cuenta contable es de 20 caracteres alfanuméricos, etc.)
- ✓ Organizar los datos para evitar redundancia e inconsistencia, a este proceso se llama Normalización
- ✓ Definir los procesos operacionales requeridos para mantener de forma efectiva los datos (Por ejemplo, calendarizar respaldos, implementar auditorias de seguridad, etc.)

Antes de entrar de lleno a la descripción de las entidades directamente involucradas con las del manejo contable debemos presentar las entidades que se encuentran involucradas en todo el DNS y que tienen que ver con el módulo contable, estas entidades forman parte de un módulo dentro del DNS al que le hemos denominado módulo de sistema. Estas entidades son las siguientes:

Debido a que nuestro modelo es un consorcio de empresas, es necesario tener un catálogo de empresas para poder identificar a que empresa corresponden de cada una de las operaciones dentro del DNS

Tabla 9.3 La tabla Compañía

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
Cia_Id	Char(4)	Primary Key	Id de la Compañía
Cia_Desc	VarChar(100)		Nombre Fiscal de la Compañía
Cia_RFC	VarChar(15)		RFC
Cia_CURP	VarChar(20)		CURP
Cia_Cemp	VarChar(20)		Cédula empresarial
Edo_Id	Char(4)	Foreing Key	Id del Estado
Municipio_Id	Char(4)	Foreing Key	Id del Municipio

Colonia_Id	Char(4)	Foreing Key	Id de la Colonia
Cia_Dir	VarChar(50)		Dirección fiscal
Cia_Tel	VarChar(25)		Telefono
Cia_Email	VarChar(40)		Email
Cia_Alias	VarChar(20)		Alias de la compañia
Cia_Status	Bit		Estado del registro

Dado que el flujo de información tiene su origen en los diferentes desarrollos o fraccionamientos y que un fraccionamiento no necesariamente corresponde a una sola región, existe la necesidad de crear dos entidades que son independientes una de la otra.

Tabla 8.4 La tabla Región

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
Reg_Id	Char(4)	Primary Key	Id de la Región
Reg_Desc	VarChar(50)		Nombre de la Región
Reg_Status	Bit		Estado del registro

Tabla 8.5 La tabla Fraccionamiento

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
Fra_Id	Char(4)	Primary Key	Id del Fraccionamiento
Fra_Desc	VarChar(100)		Nombre del Fraccionamiento
Edo_Id	Char(4)	Foreing Key	Id del Estado
Municipio_Id	Char(4)	Foreing Key	Id del Municipio
Colonia_Id	Char(4)	Foreing Key	Id de la Colonia
Fra_Dir	VarChar(50)		Dirección
Fra_Tel	VarChar(25)		Telefono
Fra_Email	VarChar(40)		Email
Fra_Ini	VarChar(20)		Iniciales del Fraccionamiento
Iva_Id			Id del IVA en el Fraccionamiento
Fra_Status	Bit		Estado del registro

El módulo contable dentro del DNS no es el único, ya que existen otros módulos como: Catálogos generales, Sembrado, Presupuestos, Compras, etc., así que existe un catálogo de Módulos del DNS

Tabla 9.6 TipoPoliza

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
TipoPoliza_Id	Char(4)	Primary Key	Id del tipo de póliza
TipoPoliza _Desc	VarChar(50)		Descripción del tipo de póliza
TipoPoliza_Status	Bit		Estado del registro

Tabla 9.7 La tabla SysMódulo

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
SysMódulo_Id	Integer	Primary Key	Id del Módulo dentro del DNS
SysMódulo_Desc	VarChar(50)		Nombre del Módulo
SysMódulo_Orden	Integer		Preferencia dentro del DNS
SysMódulo_Status	Bit		Estado del registro

El último catálogo a revisar es el de Usuarios del sistema que guarda la información necesaria y que sirve para auditar cada operación dentro del DNS.

Tabla 9.8 La tabla SysUsuario

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
SysUser_Id	Integer	Primary Key	Id del Usuario
SysUser_Usuario	Varchar(10)		Login del Usuario
SysUser_Nombre	VarChar(50)		Nombre
SysUser_ApePat	VarChar(25)		Apellido Paterno
SysUser_ApeMat	VarChar(25)		Apellido Materno
SysUser_OldPwd	VarChar(15)		Password anterior
SysUser_NewPwd	VarChar(15)		Password actual
SysUser_Status	Bit		Estado del registro

Toda operación dentro de una organización debe reflejarse en la contabilidad. La contabilidad se ayuda de cuentas contables para definir rubros en los cuales recaen las operaciones dependiendo de su naturaleza, por esto contamos con un catálogo de cuentas contables para todas las empresas dentro del consorcio. Dicho catálogo está definido de la siguiente manera:

Tabla 9.9 La tabla ContaCtaContable

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
ContaCta_Id	Char(20)	Primary Key	Id de la cuenta contable
ContaCta_Desc	Varchar(150)		Nombre de la cuenta contable

ContaCta_Categoría	Char(1)		Define si la cuenta es acumulativa "A" o es de detalle "D"
ContaCta_Activa	Bit		Define si la cuenta está activa
ContaCta_FechaAlta	DateTime	El default es la fecha del servidor	Fecha en la fue dada de alta
ContaCta_FechaCambio	DateTime		Fecha en la que sufrio modificación
SysUser_Id	Integer	Foreing Key	Id del usuario

Nótese que existe un campo que define la categoría de una cuenta contable, esto es debido a que dentro de los rubros que representan las cuentas contables existen rubros que abarcan a otros y estos a su vez son abarcados por otros y así sucesivamente. De esta forma es posible conocer el balance de una empresa al verificar los rubros de más alto nivel.

Debido al manejo contable, es necesario conocer un dato extra para cada operación dentro de un DNS. Este dato es el periodo contable en el que la operación tiene su aplicación.

Tabla 9.10 La tabla ContaPeriodo

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
Cia_id	Char(4)	Primary Key	Id de la Compañía dentro del grupo
Periodo_Id	Char(6)	Primary Key	Periodo Contable de la Compañía
ContaPer_Contabilidad	Char(6)		Periodo de la Contabilidad
ContaPer_CxP	Char(6)		Periodo de CxP
ContaPer_Status	Bit		Estado del registro

Para las operaciones que se dan dentro de un DNS, existen diferentes estados que tienen que ver con si estas operaciones son válidas o no, si serán tomadas en cuenta a futuro, etc. Por esto se requiere de un catálogo de estados de los documentos contables.

Tabla 9.11 La tabla ContaEdoDocto

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
ContaEdoDoc_Id	Char(4)	Primary Key	Id del Estado del Documento
ContaEdoDoc_Desc	VarChar(50)		Nombre del Estado del Documento
ContaEdoDoc_Status	Bit		Estado del registro

Una póliza contable es un documento que detalla los diferentes movimientos que son resultado de una operación, como ejemplo tenemos el pago de una nómina, la compra de un bien o un servicio, la venta de un producto terminado, etc. El objetivo principal del manejo contable es permitir a los usuarios las altas, bajas y cambios de una póliza contable, para o cual analizaremos las entidades involucradas en este proceso.

Tabla 9.12 La tabla ContaPoliza

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
Cia_id	Char(4)	Primary Key	Id de la Compañía dentro del grupo
Reg_id	Char(4)	Primary Key	Id de la Región a la que pertenece la compañía
Fra_Id	Char(4)	Primary Key	Id del Centro de Costos
TipoPoliza_Id	DateTime	Primary Key	Tipo de Póliza, Diario, Ingreso, Egreso, etc.
Periodo_Id	Char(6)	Primary Key	Periodo Contable en el que se aplica la póliza
ContaPol_Id	Integer	Primary Key	Número de póliza
ContaPol_Concepto	VarChar(150)		Concepto de la póliza
ContaPol_FechaCon	DateTime		Fecha de aplicación contable
ContaPol_FechaAlta	DateTime	El default es la fecha del servidor	Fecha de ingreso al sistema
ContaPol_FechaCambio	DateTime		Fecha en que sufrió cambio
ContaPol_TotalPol	Currency		Total de la póliza
ContaEdoDoc_Id	Char(4)	Foreing Key	Estado de la póliza, Balanceada, Cancelada, etc.
SysMódulo_Id	Integer	Foreing Key	Módulo dentro del DNS que ocasionó esta póliza contable
SysUser_Id	Integer	Foreing Key	Id del usuario

En esta tabla podemos ver la presencia de campos que permiten conocer a que empresa pertenece esta póliza, a que centro de costo se aplica la operación, en que fecha y en que periodo contable se hace la aplicación, así como el monto total de la operación. Vale la pena hacer mención la presencia del campo que define el módulo fuente de la operación, ya que de esta forma es posible saber de que módulo proviene esta operación, lo que significa que las pólizas contables pueden ser generadas

automáticamente como resultado de una operación en otro módulo dentro del DNS, por ejemplo una cancelación de un cliente en el módulo de Cobranza significa una póliza de egreso.

El registro de una operación, como lo hemos mencionado, se hace definiendo en que rubros recae, el detalle de tal definición se encuentra en los movimientos que constituyen a una póliza contable

Tabla 9.13 La tabla ContaPolizaDet

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
Cia_id	Char(4)	Primary Key	Id de la Compañía dentro del grupo
Reg_id	Char(4)	Primary Key	Id de la Región a la aue pertenece la compañía
Fra_Id	Char(4)	Primary Key	Id del Centro de Costos
TipoPoliza_Id	DateTime	Primary Key	Tipo de Póliza, Diario, Ingreso, Egreso, etc.
Periodo_Id	Char(6)	Primary Key	Periodo Contable en el que se aplica la póliza
ContaPol_Id	Integer	Primary Key	Número de póliza
ContaPol_NoMov	Integer		Número de movimiento
ContaCta_Id	Char(20)	Foreing Key	Id de la cuenta
ContaPol_Concepto	VarChar(150)		Concepto de la póliza
ContaPol_Debe	Currency		Cargo del movimiento
ContaPol_Haber	Currency		Abono del movimiento

Aquí lo único notable es que esta tabla es un *"child"* de ContaPoliza

Tabla 9.14 La tabla ContaHistorialCta

NOMBRE DEL CAMPO	TIPO DE DATO	ATRIBUTOS Y RELACIONES	DESCRIPCIÓN
ContaCta_Id	Char(20)	Primary Key	Id de la cuenta
ContaHC_AnoFiscal	Integer	Primary Key	Año fiscal de la contabilidad
ContaHC_Bal01	Currency		Saldo Inicial del Periodo 01
ContaHC_Debe01	Currency		Total de cargos del periodo 01
ContaHC_Haber01	Currency		Total de abonos del periodo 01

ContaHC_Bal02	Currency	Saldo Inicial del Periodo 02
ContaHC_Debe02	Currency	Total de cargos del periodo 02
ContaHC_Haber02	Currency	Total de abonos del periodo 02
ContaHC_Bal03	Currency	Saldo Inicial del Periodo 03
ContaHC_Debe03	Currency	Total de cargos del periodo 03
ContaHC_Haber03	Currency	Total de abonos del periodo 03
ContaHC_Bal04	Currency	Saldo Inicial del Periodo 04
ContaHC_Debe04	Currency	Total de cargos del periodo 04
ContaHC_Haber04	Currency	Total de abonos del periodo 04
ContaHC_Bal05	Currency	Saldo Inicial del Periodo 05
ContaHC_Debe05	Currency	Total de cargos del periodo 05
ContaHC_Haber05	Currency	Total de abonos del periodo 05
ContaHC_Bal06	Currency	Saldo Inicial del Periodo 06
ContaHC_Debe06	Currency	Total de cargos del periodo 06
ContaHC_Haber06	Currency	Total de abonos del periodo 06
ContaHC_Bal07	Currency	Saldo Inicial del Periodo 07
ContaHC_Debe07	Currency	Total de cargos del periodo 07
ContaHC_Haber07	Currency	Total de abonos del periodo 07
ContaHC_Bal08	Currency	Saldo Inicial del Periodo 08
ContaHC_Debe08	Currency	Total de cargos del periodo 08
ContaHC_Haber08	Currency	Total de abonos del periodo 08
ContaHC_Bal09	Currency	Saldo Inicial del Periodo 09
ContaHC_Debe09	Currency	Total de cargos del periodo 09
ContaHC_Haber09	Currency	Total de abonos del periodo 09
ContaHC_Bal10	Currency	Saldo Inicial del Periodo 10
ContaHC_Debe10	Currency	Total de cargos del periodo 10
ContaHC_Haber10	Currency	Total de abonos del periodo 10
ContaHC_Bal11	Currency	Saldo Inicial del Periodo 11
ContaHC_Debe11	Currency	Total de cargos del periodo 11

ContaHC_Haber11	Currency	Total de abonos del periodo 11
ContaHC_Bal12	Currency	Saldo Inicial del Periodo 12
ContaHC_Debe12	Currency	Total de cargos del periodo 12
ContaHC_Haber12	Currency	Total de abonos del periodo 12
ContaHC_Bal13	Currency	Saldo Final del Periodo 12

El diagrama entidad-relación que ilustra el modelo de datos para el manejo de pólizas contables se muestra a continuación:

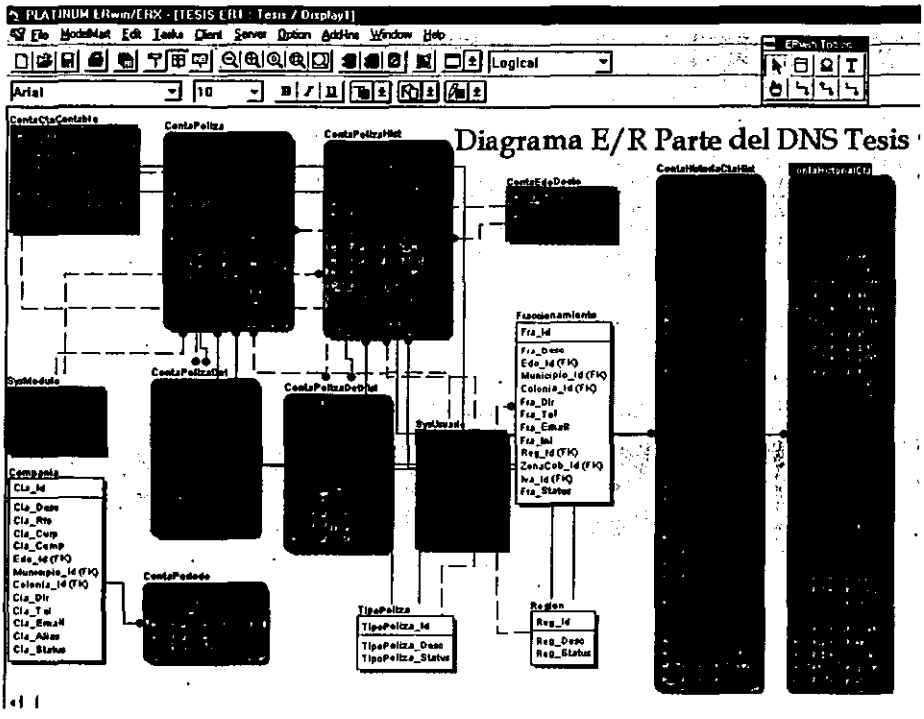


Figura 9.1 Diagrama Entidad/Relación

Para la creación del diagrama entidad-relación nos ayudamos de Erwin, una herramienta CASE. Esta herramienta, una vez definidas las entidades y las relaciones entre éstas, nos permite generar un script que podemos ejecutar directamente en la consola del Query Analizer de SQL Server.

Esta forma de crear la representación física del modelo de datos no única. Erwin permite la creación de tablas dentro de una base de datos de diversas maneras que

incluyen el uso objetos de acceso a datos como DAO[2] y ODBC *DataSources*[3]. Nosotros nos permitimos recomendar el método de la generación de un script porque nos permite visualizar cuales son las operaciones que se llevaran a cabo para la creación de las tablas, la asignación de los tipos de datos, la definición de *constraints*[4],etc., ya que nuevamente para la implementación física de un modelo de datos existen diferentes caminos, que dependen del manejador de base de datos que se utilice, por ejemplo, un *constraint* se puede definir sobre un campo al momento de la creación de la tabla o se puede crear un *constraint* de manera independiente y una vez creada la tabla se liga el *constraint* al campo.

Además utilizando el método de la creación de un script es posible modificarlo y documentarlo para que forme parte de la documentación del sistema, más aún, una vez que tenemos el script final de nuestro modelo de datos podemos modificarlo de tal manera que en el se incluyan las sentencias necesarias para la creación de los dispositivos en los que residirá la base de datos, la creación de la base de datos misma, instrucciones de configuración de la base de datos como el conjunto de caracteres, modificar los mensajes de error debido a la violación de algún constraint, regla o ejecución de un trigger, etc.
Para implementar el modelo de datos:

1. Desde Erwin, del menú Tasks → Forward Engineer/Eschema Generation...
   Indique la ruta en donde se salvará el archivo que contiene el script para la generación del modelo de datos.

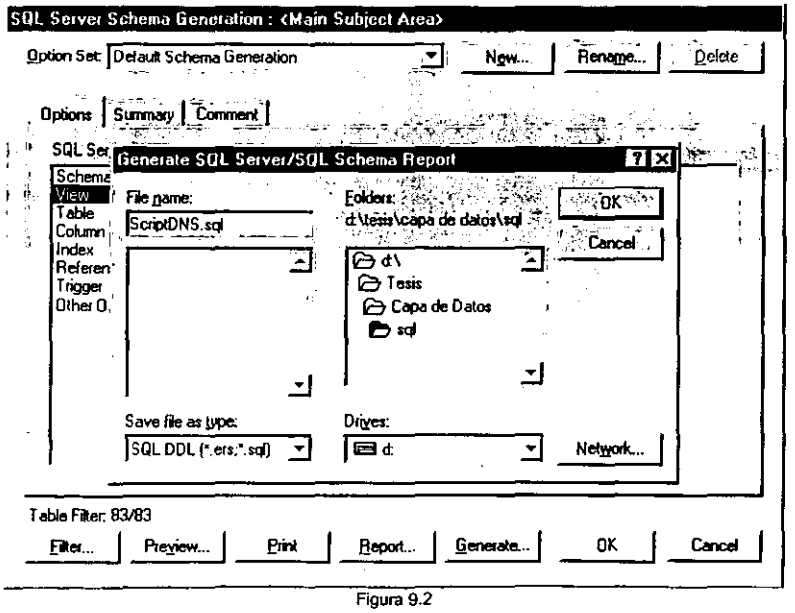


Figura 9.2

[2] Data Access Objects
[3] Fuentes o repositorios de datos
[4] Restricciones de bases de datos

2. Accese al Query Analizer de SQL Server ...
   Seleccione el servidor de base de datos en donde ejecutará el script

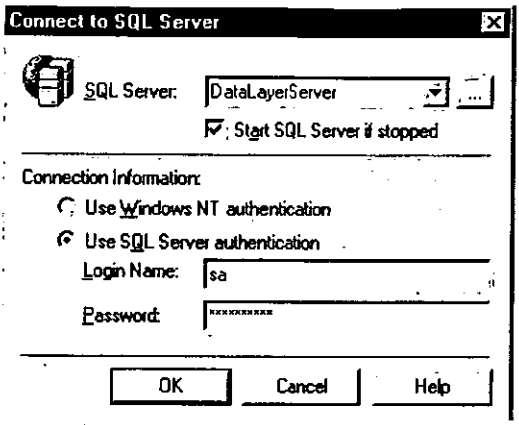


Figura 9.3

3. Del menú File → Open ...
   Especifique la ruta donde se encuentra el script generado por Erwin
4. Una vez abierto el script, ejecute el menú Query → Execute ...

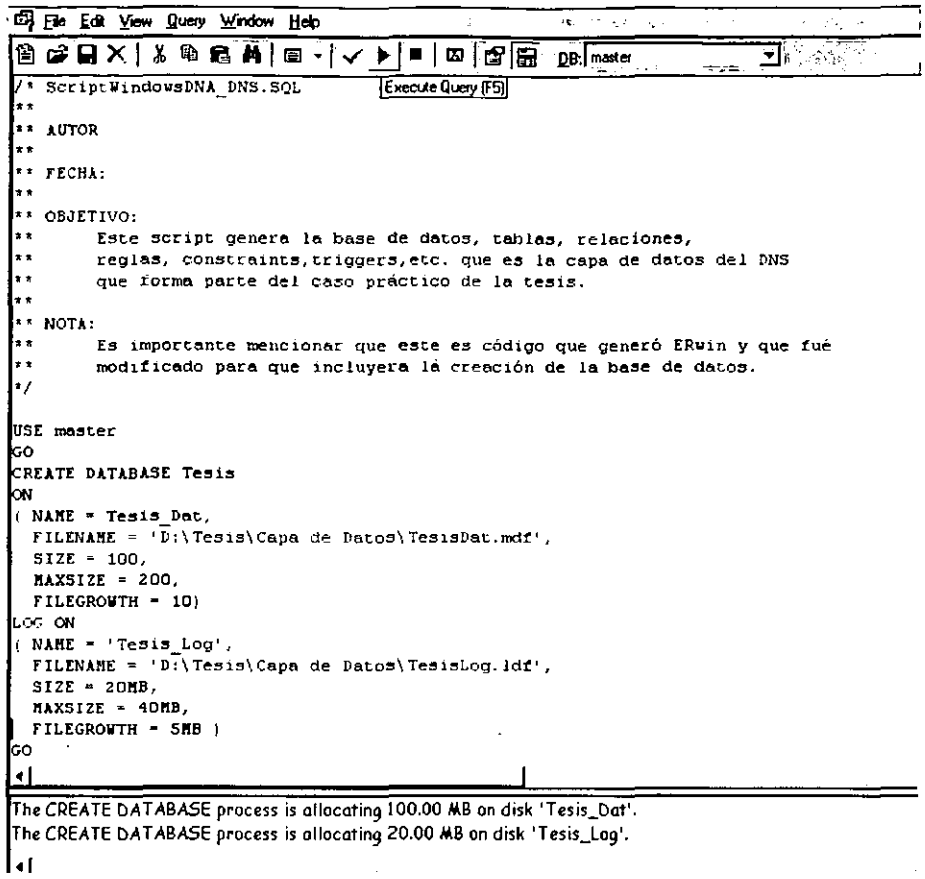


Figura 9.4

Terminada la ejecución el modelo de datos y propiamente la capa de datos estará implementada.

PASO 3. DEFINICIÓN DE LAS REGLAS DE NEGOCIOS (CAPA DE NEGOCIOS)

La parte de la Capa de Negocios del DNS que presentaremos, se refiere al manejo contable y básicamente involucra el manejo de las cuentas contables y de las pólizas

La definición de las operaciones de la capa de negocios implica:

- ✓ Identificar las operaciones que reflejan los procesos de negocios que se pretenden automatizar con la aplicación.
- ✓ Definir y publicar la sintaxis para cada una de las peticiones a operaciones de negocios.

Identificar las operaciones que reflejan los procesos de negocios

El manejo de las cuentas y pólizas contables define operaciones de negocio que reflejan el proceso de mantenimiento de las cuentas y en el registro y seguimiento de la contabilidad. Estas operaciones se pueden organizar en los siguientes grupos:

- ✓ Registro de cuentas
- ✓ Actualización de cuentas
- ✓ Eliminación de cuentas
- ✓ Mantenimiento del Saldo
- ✓ Registro de pólizas contables
- ✓ Actualización de pólizas contables
- ✓ Eliminación de pólizas contables

La tablas siguientes describen cada uno de los grupos de operaciones definidas para el mantenimiento de las cuentas y pólizas contables:

Tabla 9.15 Operaciones definidas para el registro de una cuenta contable

OPERACIÓN	DESCRIPCIÓN
UsuarioTienePermiso	Verifica que el usuario tenga permiso de crear una cuenta contable
QuitaCar	Elimina el formato con el cual es presentada la cuenta al usuario
EsCtadeCia	Verifica si la cuenta pertenece a la Compañía
CtaParentValida	Verifica si la cuenta tiene una cuenta acumulativa
FechaAct	Obtiene la fecha actual en el servidor de base de datos
ContaCtaContable	Inserta la cuenta al catálogo
DfneCategCtaParent	Define la posición de la cuenta dentro del catálogo
ObtenAñoInicial	Obtiene el año inicial de la contabilidad
ObtenAñoFinal	Obtiene el año actual de la contabilidad
ContaHistorialCta	Inserta la cuenta para llevar su historial contable
ContaCtaContableRegion	Inserta la cuenta en las regiones especificadas por el usuario

Tabla 9.16 Operaciones definidas para la actualización de cuentas contables

OPERACIÓN	DESCRIPCIÓN
UsuarioTienePermiso	Verifica que el usuario tenga permiso de modificar una cuenta contable
QuitaCar	Elimina el formato con el cual es presentada la cuenta al usuario
EsCtadeCia	Verifica si la cuenta pertenece a la Compañía
FechaAct	Obtiene la fecha actual en el servidor de base de datos
ContaCtaContable	Actualiza la cuenta en el catálogo
ContaCtaContableRegion	Actualiza la cuenta en las regiones especificadas por el usuario

ObtenAñoInicial	Obtiene el año inicial de la contabilidad
ObtenAñoFinal	Obtiene el año actual de la contabilidad
ObtenSaldoIni	Obtiene el saldo a la fecha para la cuenta
ContaHistorialCtaHistSaldos	Actualiza el saldo de la cuenta en su historial contable

Tabla 9.17 Operaciones definidas para la eliminación de cuentas contables

OPERACIÓN	DESCRIPCIÓN
UsuarioTienePermiso	Verifica que el usuario tenga permiso de eliminar una cuenta contable
QuitaCar	Elimina el formato con el cual es presentada la cuenta al usuario
EsCtadeCia	Verifica si la cuenta pertenece a la Compañía
TieneMovimientos	Verifica si la cuenta puede ser borrada
DfneCategCtaParent	Define la posición de la cuenta acumulativa dentro del catálogo
ContaHistorialCta	Borra el historial de la cuenta
ContaHistorialCtaHist	Borra el historial de la cuenta
ContaCtaContableRegion	Borra la cuenta de las regiones en donde este asignada
ContaCtaContable	Borra la cuenta del catálogo de cuentas

Tabla 9.18 Operaciones definidas para el registro de pólizas contables

OPERACIÓN	DESCRIPCIÓN
UsuarioTienePermiso	Verifica que el usuario tenga permiso de crear una póliza contable
RevisaExistencia	Revisa si la póliza contable ya se encuentra en el sistema
ObtenDetalleCuadrado	Verifica la consistencia del detalle de la póliza contable
ObtenNumeroPoliza	Obtiene el número de la póliza
FechaAct	Obtiene la fecha actual en el servidor de base de datos
InsertContaPoliza	Inserta la póliza a la base de datos
InsertDetalle	Inserta y contabiliza el detalle de la póliza a la base de datos

Tabla 9.19 Operaciones definidas para la actualización de pólizas contables

OPERACIÓN	DESCRIPCIÓN
UsuarioTienePermiso	Verifica que el usuario tenga permiso de actualizar una póliza contable
PuedeModificarDocto	Verifica si la póliza contable puede ser modificada
ObtenDetalleCuadrado	Verifica la consistencia del detalle de la póliza contable
FechaAct	Obtiene la fecha actual en el servidor de base de datos
DeContabilizaPoliza	Reversa el asiento contable anterior de la póliza
UpdateContaPoliza	Actualiza el encabezado de la póliza
DeleteDetalle	Borra el detalle anterior de la póliza
InsertDetalle	Inserta y contabiliza el detalle de la póliza a la base de datos

Tabla 9.20 Operaciones definidas para la eliminación de las pólizas contables

OPERACIÓN	DESCRIPCIÓN
UsuarioTienePermiso	Verifica que el usuario tenga permiso de eliminar una póliza contable
PuedeModificarDocto	Verifica si la póliza contable puede ser eliminada
FechaAct	Obtiene la fecha actual en el servidor de base de datos
DeleteContaPoliza	Borra lógicamente la póliza de la contabilidad
DeContabilizaPoliza	Reversa el asiento contable de la póliza

Definir y publicar la sintaxis para cada una de las peticiones a operaciones de negocios.

Una vez definidas las diferentes operaciones de negocios, es necesario definir una sintaxis para el acceso a las operaciones de negocios, de tal forma que los clientes conozcan el formato de como deben dirigirse para poder ejecutar la operación deseada. Para nuestro caso hemos decidido emplear interfaces que contengan toda la información que necesita el cliente para ejecutar la operación de negocio deseada.

Al definir las interfaces se debe tener especial cuidado en que toda la información requerida para ejecutar una operación de negocio se pueda llevar a cabo con un mínimo de viajes entre el cliente y el servidor.

Nuestra aplicación define 3 interfaces COM con 6 métodos que describen la información que es requerida para llevar a cabo las diferentes operaciones de negocio:

Las interfaces son :

- ✓ bus_Contabilidad.Insert
- ✓ bus_Contabilidad.Updata
- ✓ bus_Contabilidad.Delete

Los métodos dentro de estas interfaces son:

- ✓ Insert.ContaCtaContable
- ✓ Update.ContaCtaContable
- ✓ Delete. ContaCtaContable
- ✓ Insert.Poliza
- ✓ Update.Poliza
- ✓ Delete:Poliza

Las siguientes tablas describen los diferentes parámetros que requieren las interfaces para llevar a cabo su trabajo.

Tabla 9.21 El método ContaCtaContable de la interfaz bus_Contabilidad.Insert

PARÁMETRO	TIPO DE DATO	DESCRIPCIÓN
SysUser_Id	String	Id del Usuario que requiere la operación

Cia_Id	String	Id de la Compañía
Reg_Id	String	Id de la Región
Fra_Id	String	Id del Fraccionamiento
ContaCta_Id	String	Id de la Cuenta Contable
ContaCta_Desc	String	Nombre de la Cuenta Contable
Reg_Ids	Variant	Arreglo de Id´s de las regiones donde aplica la Cuenta Contable
ContaHC_Bal01	String	Saldo inicial de la Cuenta

Tabla 9.22 El método ContaCtaContable de la interfaz bus_Contabilidad.Update

PARÁMETRO	TIPO DE DATO	DESCRIPCIÓN
SysUser_Id	String	Id del Usuario que requiere la operación
Cia_Id	String	Id de la Compañía
Reg_Id	String	Id de la Región
Fra_Id	String	Id del Fraccionamiento
ContaCta_Id	String	Id de la Cuenta Contable
ContaCta_Desc	String	Nombre de la Cuenta Contable
ContaHC_Bal01	String	Saldo inicial de la Cuenta
ContaCta_Activa	String	Status de la cuenta

Tabla 9.23 El método ContaCtaContable de la interfaz bus_Contabilidad.Delete

PARÁMETRO	TIPO DE DATO	DESCRIPCIÓN
SysUser_Id	String	Id del Usuario que requiere la operación
Cia_Id	String	Id de la Compañía
Reg_Id	String	Id de la Región
Fra_Id	String	Id del Fraccionamiento
ContaCta_Id	String	Id de la Cuenta Contable

Tabla 9.24 El método Poliza de la interfaz bus_Contabilidad.Insert

PARÁMETRO	TIPO DE DATO	DESCRIPCIÓN
SysUser_Id	String	Id del Usuario que requiere la operación
Cia_Id	String	Id de la Compañía
Reg_Id	String	Id de la Región
Fra_Id	String	Id del Fraccionamiento
TipoPoliza_Id	String	Id del tipo de documento contable (Póliza de Diario, Ingreso, Egreso, etc.)
Periodo_Id	String	Id del periodo contable
ContaPol_Concepto	String	Concepto de la póliza contable
ContaPol_FechaCon	String	Fecha de aplicación contable
arr_Detalle	Variant	Arreglo que contiene el detalle de los movimientos contables
RevisaExistencia	String	Determina si se verificará la existencia de la póliza contable en la contabilidad

Tabla 9.25 El método Poliza de la interfaz bus_Contabilidad.Update

PARÁMETRO	TIPO DE DATO	DESCRIPCIÓN
SysUser_Id	String	Id del Usuario que requiere la operación
Cia_Id	String	Id de la Compañía
Reg_Id	String	Id de la Región
Fra_Id	String	Id del Fraccionamiento
TipoPoliza_Id	String	Id del tipo de documento contable (Póliza de Diario, Ingreso, Egreso, etc.)
Periodo_Id	String	Id del periodo contable
ContaPol_Id	String	Id de la póliza contable a actualizar
ContaPol_Concepto	String	Concepto de la póliza contable a actualizar
ContaPol_FechaCon	String	Fecha de aplicación contable a actualizar
arr_Detalle	Variant	Arreglo que contiene el detalle de los nuevos movimientos contables

Tabla 9.26 El método Poliza de la interfaz bus_Contabilidad.Delete

PARÁMETRO	TIPO DE DATO	DESCRIPCIÓN
SysUser_Id	String	Id del Usuario que requiere la operación
Cia_Id	String	Id de la Compañía
Reg_Id	String	Id de la Región
Fra_Id	String	Id del Fraccionamiento
TipoPoliza_Id	String	Id del tipo de documento contable (Póliza de Diario, Ingreso, Egreso, etc.)
Periodo_Id	String	Id del periodo contable
ContaPol_Id	String	Id de la póliza contable a eliminar

PASO 4. CREACIÓN DE LOS OBJETOS EJECUTANTES (EXECUTANTS)

Una vez que las operaciones de negocio y las respectivas sintaxis de nuestro DNS han sido definidas, el siguiente paso es crear los ejecutantes que a final de cuentas llevarán a cabo las operaciones de negocio sobre la base de datos. Dado que los objetos ejecutantes son los encargados de reflejar las operaciones de negocio en la base de datos, en estos objetos se tiene la libertad de decidir que medios emplearemos para modificar la base de datos. Algunas aplicaciones Windows DNA utilizan *stored procedures* para manipular y actualizar las bases de datos, para nuestro caso hemos optado por definir sentencias SQL para la manipulación de la base de datos.

Siguiendo la misma estructura que se definió para la creación de las interfaces de las operaciones de negocios, es posible generar un componente que se encargue de los accesos a la base de datos ofreciendo transparencia en el manejo de los datos a los componentes que hagan uso de sus métodos.

Así tenemos un componente, que tiene tres interfaces y cada una de éstas contienen métodos que se encargan del acceso directo a la base de datos. El componente se

llama dat_Contabilidad y las interfaces que expone son Insert, Update y Delete. Los métodos de estas interfaces se muestran en las siguientes tablas.

Tabla 9.27 Métodos en la interfaz Insert del componente de datos dat_Contabilidad

MÉTODO	DESCRIPCIÓN
ContaCtaContable	Inserta un registro en la tabla ContaCtaContable
ContaPeriodo	Inserta un registro en la tabla ContaPeriodo
ContaEdoDocto	Inserta un registro en la tabla ContaEdoDocto
ContaPoliza	Inserta un registro en la tabla ContaPoliza
ContaPolizaDet	Inserta un registro en la tabla ContaPolizaDet
ContaHistorialCta	Inserta un registro en la tabla ContaHistorialCta

Tabla 9.28 Métodos en la interfaz Update del componente de datos dat_Contabilidad

MÉTODO	DESCRIPCIÓN
ContaCtaContable	Actualiza un registro en la tabla ContaCtaContable
ContaCtaContableRegion	Actualiza un registro en la tabla ContaCtaContableRegion
ContaPoliza	Actualiza un registro en la tabla ContaPoliza
ContaPolizaDet	Actualiza un registro en la tabla ContaPolizaDet
ContaHistorialCtaCA	Actualiza uno o varios registros en los campo de cargos y abonos (ContaHC_DebeN,ContaHC_HaberN) en la tabla ContaHistorialCta dependiendo de la categoria de la cuenta que se este actualizando y del periodo contable
ContaHistorialCtaSaldos	Actualiza uno o varios registros en los campo saldos iniciales (ContaHC_BalN) en la tabla ContaHistorialCta dependiendo de la categoria de la cuenta que se este actualizando y del periodo contable

Tabla 9.29 Métodos en la interfaz Delete del componente de datos dat_Contabilidad

MÉTODO	DESCRIPCIÓN
ContaCtaContable	Borra un registro en la tabla ContaCtaContable
ContaPeriodo	Borra un registro en la tabla ContaPeriodo
ContaEdoDocto	Borra un registro en la tabla ContaEdoDocto
ContaPoliza	Borra un registro en la tabla ContaPoliza
ContaPolizaDet	Borra un registro en la tabla ContaPolizaDet
ContaHistorialCta	Borra un registro en la tabla ContaHistorialCta

Cada uno de los métodos de las interfaces Insert, Update y Delete tiene una lista de parámetros que requiere para poder ejecutar sus operaciones sobre la base de datos, en ocasiones esta lista de parámetros contiene parámetros opcionales, que permiten utilizar el método de tal forma que solo inserte, actualice o elimine registros de acuerdo a los parámetros proporcionados. Así, por ejemplo, un objeto de Update puede actualizar sólo algunos de los campos de un registro particular de la tabla para la cual fue diseñado.

El procedimiento para la generación de los componentes ejecutantes es el siguiente:

1. Desde Visaul Basic, del menú File New Project ...
   Seleccione Activex DLL y de click en OK

2. Del menú de Project Project Properties ...
   Escriba en el nombre del proyecto dat_Contabilidad

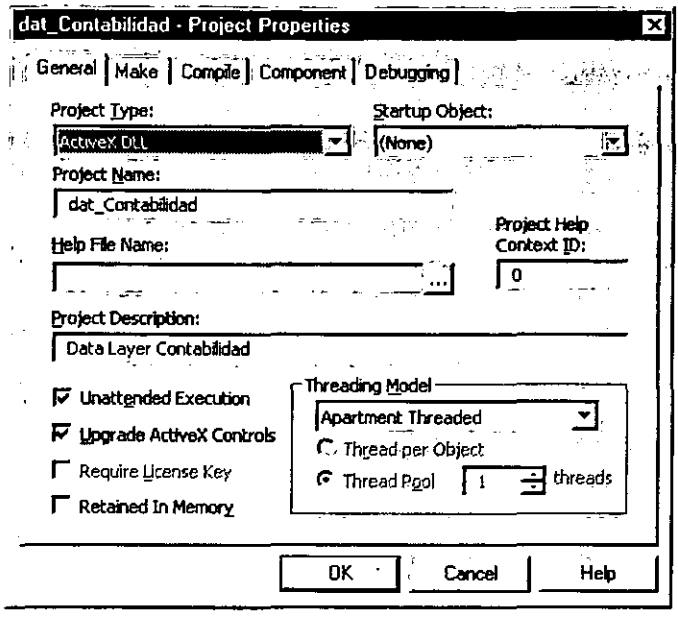


Figura 9.5

3. Desde el menú de Project References ...
   Marque la referencia a:
   Activex Data Objects 2.1 Library
   Transaction Server Type Library

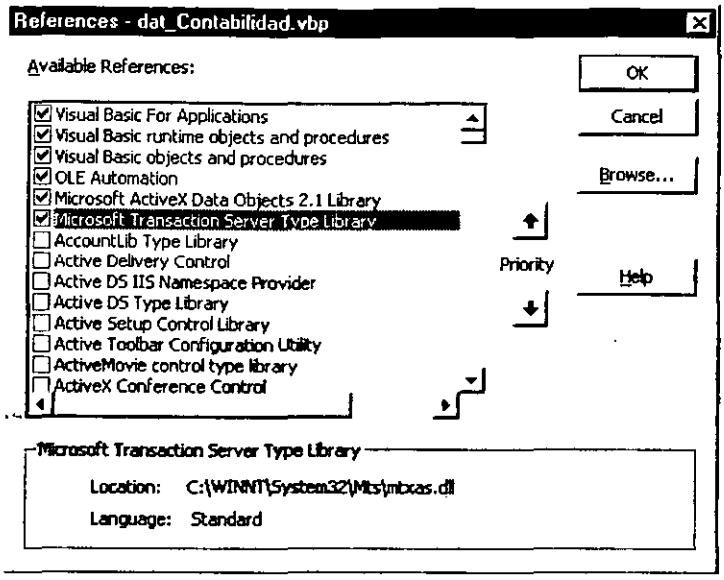


Figura 9.6

4. Desde el menú de Project Add Class Module ...
   Adicione tres nuevos módulos clase y nómbrelos como Insert, Update y Delete
   Adicione un módulo estándar para la declaración de variables globales

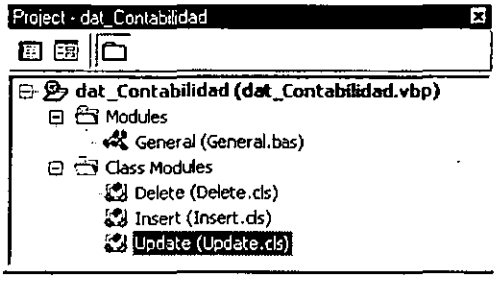


Figura 9.7

5. Codifique en cada uno de los módulo clase los métodos mencionados anteriormente para las operaciones directas sobre la base de datos.

(General) | ContaCtaContable

```
Public Sub ContaCtaContable(ByVal ContaCta_Id As String)
' Autor:
'
' Fecha:
'       |
' Parametros:
'       ContaCta_Id = Id de la Cuenta
' Objetivo:
'       Eliminar físicamente el registro de la base de datos

Dim adoCon As ADODB.Connection
Dim adoErr As ADODB.Error
Dim objContext As MTxAS.ObjectContext

On Error GoTo errHandler

    Set objContext = GetObjectContext

    'Prepara la sentencia
    str_SQL = "Delete From ContaCtaContable Where " & _
    "ContaCta_Id='" & ContaCta_Id & "'"

    Set adoCon = CreateObject("ADODB.Connection")
    adoCon.ConnectionString = adoConString
    adoCon.Open
    adoCon.Execute str_SQL
    adoCon.Close
    Set adoCon = Nothing

    'Acepta la transacción
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
```

Figura 9.8

El código completo de las interfaces de acceso a datos se encuentra en el apéndice C "Listado del Caso Práctico."

6. En la ventana de propiedades del proyecto ...
   Poner el valor de la propiedad MTSTransactionMode a "2-RequiresTransaction" para cada una de las clases, esto define que el componente siempre va a formar parte de la transacción en que esta el componente que lo llama, en caso de que el componente que lo llama no cuente con una transacción, entonces se creará una transacción.

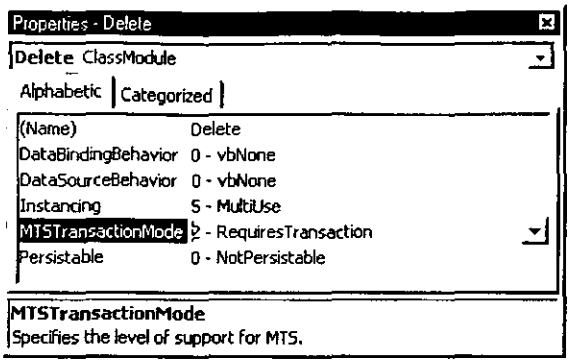


Figura 9.9

Por último, es necesario compilar el componente a un ActiveX DLL y el nombre será dat_Contabilidad.dll.

dat_Contabilidad.dll es ahora un componente que expone tres interfaces: Insert, Update y Delete que pueden ser accesadas como: dat_Contabilidad.Insert, dat_Contabilidad.Update y dat_Contabilidad.Delete. Cada una de las interfaces expone métodos, que pueden ser accesados instanciando una interfaz del componente. Un ejemplo se presenta a continuación:

```
set objDelete = Createobject("dat_Contabilidad.Delete")
```

así, el método ContaCtaContable de la interfaz Delete se puede accesar de la siguiente manera:

```
objDelete.ContaCtaContable varContaCta_Id
```

PASO 5. CREACIÓN DE LOS OBJETOS EMISARIOS (EMISSARIES)

La existencia de los objetos emisario es primordial para que el cliente puede formatear perfectamente las peticiones de las operaciones de negocios. La creación de los objetos emisarios incluye:

- ✓ Proporcionar información adicional que es necesaria para que las peticiones a los objetos ejecutantes sean válidas.
- ✓ Pre-validar cada petición del cliente lo más posible, antes de enviar la petición a un objeto ejecutante para su ejecución.

Proporcionar información adicional que es necesaria para que las peticiones a los objetos ejecutantes sean válidas.

Los objetos emisario son los responsables de ayudar a los clientes del DNS a enviar perfectamente formateadas las peticiones a los objetos ejecutantes para llevar a cabo las diferentes operaciones de negocio. Esta responsabilidad incluye el proporcionar cualquier información suplementaria que pueda necesitar la aplicación peticiones

válidas a operaciones de negocio. Para identificar tal información suplementaria, los desarrolladores deben analizar cada petición. Por ejemplo, el ingreso de una póliza contable requiere incluir, entre otros datos, el identificador de la compañía, la región a la que pertenece, el fraccionamiento en el que se genera la póliza, el tipo de póliza contable, el periodo en el que se contabilizará, el número de póliza, la fecha contable, la fecha de alta y la fecha de ultima modificación. Todos los datos antes mencionados, el usuario podría proporcionarlos, pero esto incrementa la probabilidad de errores en el sistema debido a una captura errónea de información. Para reducir la probabilidad de errores, diseñamos a los objetos emisarios para que se encarguen de determinar los más datos posibles para dejar que el usuario proporcione la información que no se puede determinar y que forma parte de la operación diaria del negocio. Por lo que un objeto emisario puede determinar a que compañía se esta accesando, con este dato y el identificador del usuario determinar la región, una vez obtenidos estos datos es tarea fácil para el objeto emisario, el determinar el periodo actual de esa compañía, por otra parte las fechas de alta y de modificación inicialmente son las mismas y éstas se determinan de acuerdo al tiempo en el servidor de base de datos, con lo que finalmente el usuario solo tiene que proporcionar el fraccionamiento, el tipo de póliza contable y la fecha contable. Por último, es también tarea del objeto emisario el determinar el consecutivo de la póliza.

Pre-validar cada petición del cliente lo más posible, antes de enviar la petición a un objeto ejecutante para su ejecución.

Para minimizar el tráfico en la red asociado con el envio y reenvio de peticiones, los objetos emisarios deben validar cada petición del cliente antes de enviarla a los objetos ejecutantes. Continuando con nuestro ejemplo de ingreso de pólizas contables, nuestros objetos emisarios deben poder validar si el usuario tiene permisos de ingresar pólizas contables, verificar que el conjunto de cuentas contables que forman parte del detalle de la póliza sea válido dentro del catálogo de cuentas contables para la compañía en cuestión, identificar si el total de cargos es igual al total de abonos, etc.

Existen diferentes formas de hacer esta validación y en general de hacer que el papel de los objetos emisarios se lleve a cabo. Los objetos emisarios hacen uso explicito de todas las reglas de negocio que hemos definido en el paso tres[5] y ahora veremos consolidadas esas reglas de negocio, siguiendo una lógica de operación. Para nuestro caso práctico hemos dividido a los objetos emisarios en 3 capas:

Una capa es la encargada de hacer todas las validaciones necesarias y de recavar la información necesaria para poder generar una operación de negocio satisfactoriamente. Este objeto llamado util_Contabilidad tiene cuatro interfaces, cada una para propósitos muy particulares:

Tabla 9.30 Interfaces del objeto emisario util_Contabilidad

INTERFAZ	HACE USO DE	PROPÓSITO
HTML	MDAC	Se encarga de generar código HTML "al vuelo" para la presentación de listas

[5] Definición de las reglas de negocio (Capa de Negocios)

		extraídas de la base de datos
Utilerías	MDAC	En esta se encuentran utilerías varias para satisfacer las necesidades de otros objetos y que no tienen que ver con el negocio.
Seguridad	MDAC	Aquí se encuentran las rutinas necesarias para validar el acceso al sistema, información sobre permisos, etc.
Negocio	dat_Contabilidad.Insert dat_Contabilidad.Update dat_Contabilidad.Delete util_Contabilidad.Utilerias util_Contabilidad.Seguridad util_Contabilidad.Negocio	Se encarga de digerir la operación del a través de métodos en los cuales se refleja a detalle cada operación del negocio

La capa de más arriba es la encargada de obtener la información directamente del usuario, es decir la capa a la cual llega cada petición de operación de negocio. Este objeto se llama bus_Contabilidad y se encarga de recibir las peticiones del cliente. Hace uso de la interfaz util_Contabilidad.Negocio para llevar a cabo la operación de negocio tiene tres interfaces

Tabla 9.31 Interfaces del objeto emisario bus_Contabilidad

INTERFAZ	HACE USO DE	DESCRIPCIÓN
bus_Contabilidad.Insert	util_Contabilidad.Seguridad util_Contabilidad.Negocio dat_Contabilidad.Insert	En esta capa se llevan a cabo todas las operaciones de registro de operaciones de negocio
bus_Contabilidad.Update	util_Contabilidad.Seguridad util_Contabilidad.Negocio dat_Contabilidad.Update	En esta capa se llevan a cabo todas las operaciones de actualización o borrado lógico de operaciones de negocio.
bus_Contabilidad.Delete	util_Contabilidad.Seguridad util_Contabilidad.Negocio dat_Contabilidad.Delete	En esta capa se llevan a cabo todas las operaciones de borrado físico de operaciones de negocio

Se presentan a continuación los pasos que se deben llevar a cabo para definir las interfaces, que los clientes utilizarán para ejecutar las diferentes operaciones de negocio definidas para el subsistema contable del DNS.

Para el caso de el componente bus_Contabilidad:

1. Desde Visaul Basic, del menú File New Project ...
   Seleccione Activex DLL y de click en OK
2. Del menú de Project Project Properties ...
   Escriba en el nombre del proyecto bus_Contabilidad

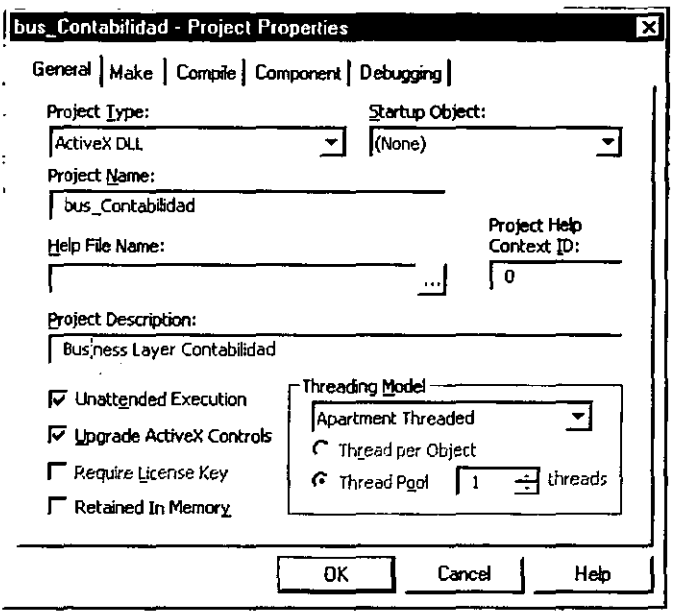


Figura 9.10

3. Desde el menú de Project References ...
   Marque la referencia a:
   Transaction Server Type Library
   Util_Contabilidad.dll (Utilerias de Contabilidad)
   Dat_Contabilidad.dll (Data Layer Contabilidd)

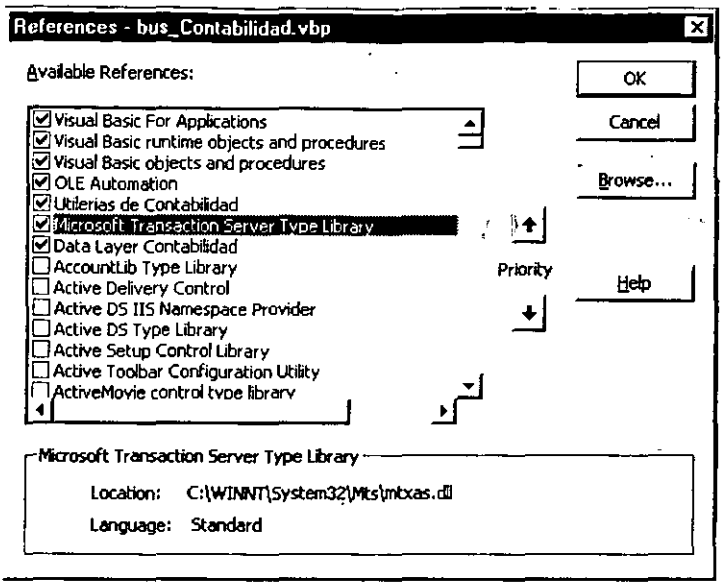


Figura 9.11

4. Desde el menú de Project Add Class Module ...
   Adicione tres nuevos módulos clase y nómbrelos como Insert, Update y Delete
   Adicione un módulo estándar para la declaración de variables globales

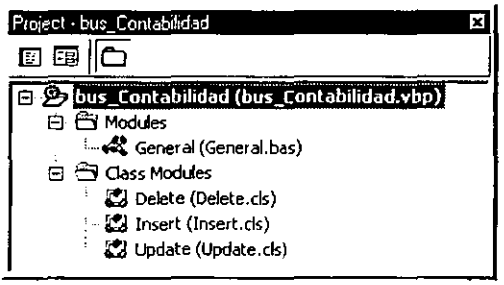


Figura 9.12

5. Codifique en cada uno de los módulos clase la declaración de los métodos mencionados anteriormente para el manejo de las operaciones de negocio.

El código completo de las interfaces utilizadas se encuentra en el apéndice C "Listado del Caso Práctico"

```
(General)                                    Poliza

Public Sub Poliza(ByVal SysUser_Id As String, ByVal Cia_Id As String, _
    ByVal Reg_Id As String, ByVal Fra_Id As String, ByVal TipoPoliza_Id As String, _
    ByVal Periodo_Id As String, ByVal ContaPol_Concepto As String, _
    ByVal ContaPol_FechaCon As String, ByRef arr_Detalle As Variant, _
    ByVal RevisaExistencia As String)
' Autor:
'
' Fecha:
'
' Parametros:
'       Cia_Id                  =   Id de la Comañia
'       Reg_Id                  =   Id de la Foranea
'       Fra_Id                  =   Id del Fraccionamiento
'       TipoPoliza_Id           =   Tipo de Poliza (IG,EG,DR,CP,TE,etc.)
'       Periodo_Id              =   Periodo contable
'       ContaPol_Concepto       =   Concepto de la Poliza
'       ContaPol_FechaCon       =   Fecha de contabilizacion
'       arr_Detalle             =   Contiene el deatalle de movimientos
'       RevisaExistencia        =   Define si se revisara la existencia de la poliza
'       SysUser_Id              =   Id del usuario
'
' Objetivo:
'       Inserta la póliza contable y hace la afectación necesaria

Dim int_Intentos As Integer
Dim objNegocios As util_Contabilidad.Negocios
Dim objUtilerias As util_Contabilidad.Utilerias
Dim str_Fecha As String
Dim cur_ContaPol_TotalPol As Currency
Dim lng_ContaPol_Id As Long
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
Intenta:
    Set objContext = GetObjectContext
```

Figura 9.13

6. En la ventana de propiedades del proyecto

   Poner el valor de la propiedad MTSTransactionMode a "4-RequiresNewTransaction" para cada una de las clases, esto define que el componente siempre va a iniciar una nueva transacción en cada llamada una de las llamadas a éste.

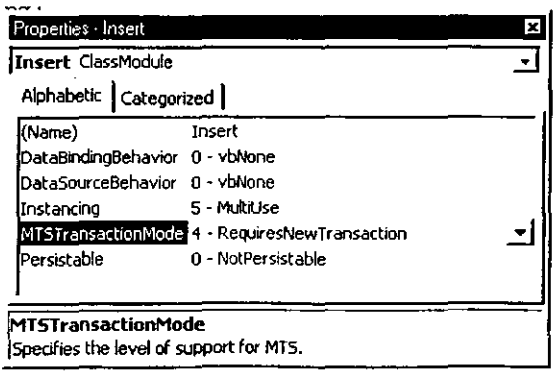


Figura 9.14

Una vez que todas las interfaces se encuentran definidas y que los métodos dentro de cada una de éstas se encuentran codificados, es necesario compilar el componente a un ActiveX DLL y el nombre será bus_Contabilidad.dll.

bus_Contabilidad.dll es ahora un componente que expone tres interfaces: Insert, Update y Delete que pueden ser accesadas como: bus_Contabilidad.Insert, bus_Contabilidad.Update y bus_Contabilidad.Delete. Cada una de las interfaces expone métodos, que pueden ser accesados instanciando una interfaz del componente. Un ejemplo se presenta a continuación:

```
set objInsert = Createobject("bus_Contabilidad.Insert")
```

así, el método Poliza de la interfaz Insert se puede accesar de la siguiente manera:

```
objInsert.Poliza varSysUser_Id,varCia_Id,varReg_Id, _
varFra_Id,varTipoPoliza_Id,varPeriodo_Id,varContaPol_Concepto,
varContaPol_FechaCon,arr_Detalle,varRevisaExistencia
```

La última capa se encuentra en el lado del cliente en forma de un control ActiveX, que esta construido para que internamente guarde las reglas de negocio que tienen que ver con la captura y modificación del detalle de los documentos contables. Este control aunque se encuentra directamente en el cliente, no forma parte de la capa de Presentación, ya que su objetivo principal es validar la entrada de información al sistema, que es una operación de negocios.

El procedimiento para crear el control Activex , es el siguiente

1. Desde Visaul Basic, del menú File New Project ...
   Seleccione Activex Control y de click en OK
2. Del menú de Project Project Properties ...
   Escriba en el nombre del proyecto Grid

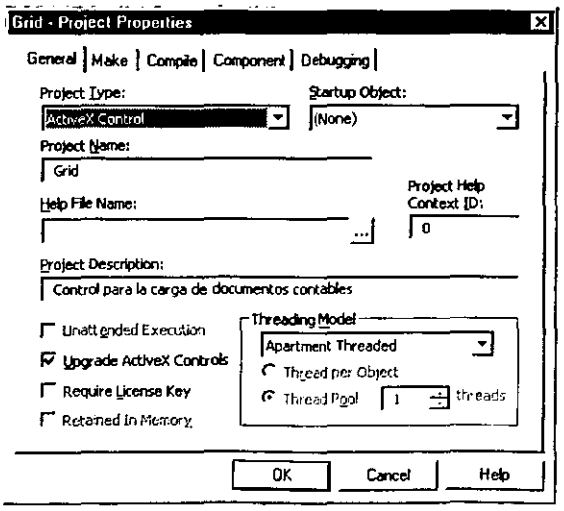


Figura 9.15

3. Desde el menú de Project Grid Properties ...
   Defina el nombre del control como UControlGrid

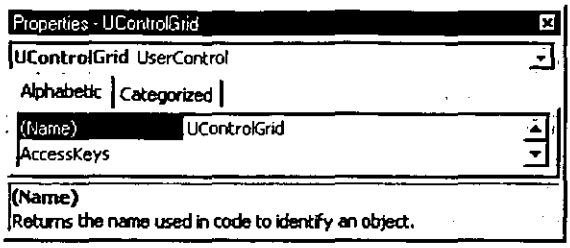


Figura 9.16

4. Desde el menú de Project References ...
   Marque la referencia a:
   Remote Data Object 2.0

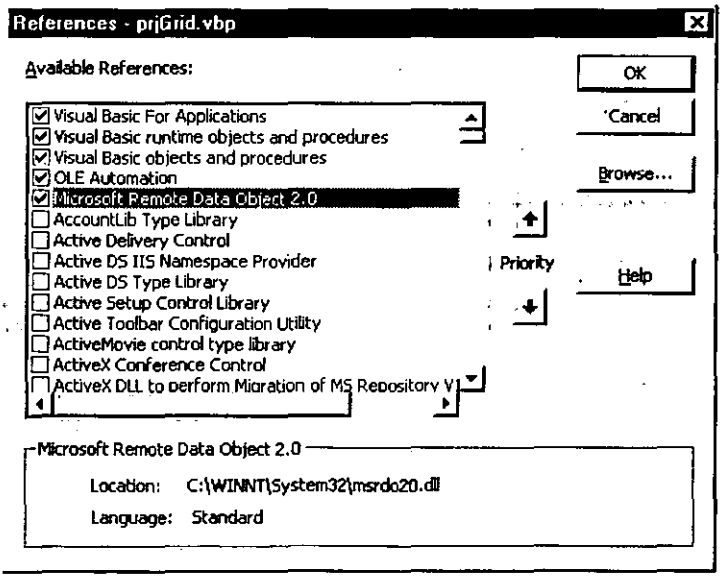


Figura 9.17

5. Desde el menú de Project Components ...
   Marque el componente:
   Remote Data Control 6.0
   Sheridan Data Grid/Combo/Drop Down 3.1

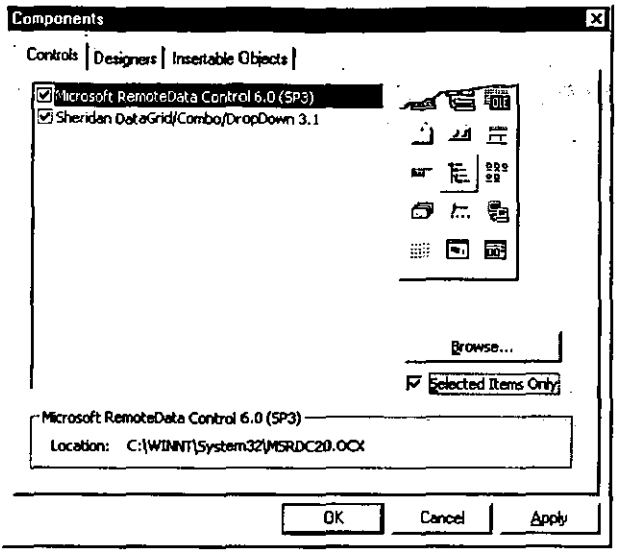


Figura 9.18

6. En el UserControl adicione los componentes:
   SSDBGrid
   SSDBDropDown
   MSRDC

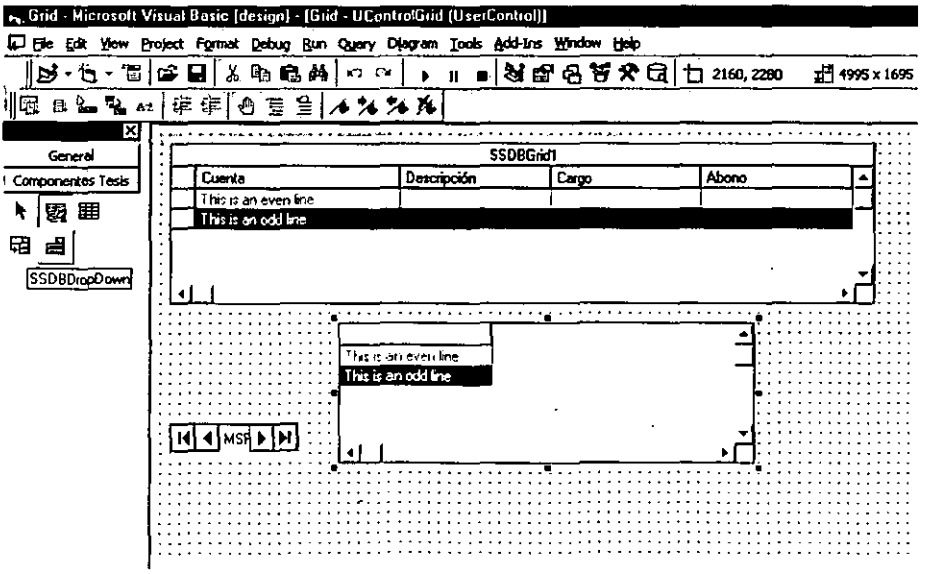


Figura 9.19

7. Codifique en la ventana de código presentado en el apéndice C "Listado del Caso Práctico"

(General) | ChecaGrid

```
Private Function ChecaGrid() As Boolean
' Autor:
' Fecha:
' Parametros:
' Objetivo:
'       Verifica que el detalle de la poliza contable que se trata de ingresar al sist
'       sea consistente, es decir que sean cuentas válidas,cargos y abonos válidos

    'Dado que el bookmark es relativo al registro en el que se encuentre
    'vamos al principio para asegurar pasar por todos los registros
    Dim var_BookMark As Variant
    Dim cur_TotalCargos As Currency
    Dim cur_TotalAbonos As Currency
    Dim int_Cont As Integer

    var_BookMark = SSDBGrid.Bookmark
    SSDBGrid.MoveFirst
    For int_Cont = 0 To SSDBGrid.Rows - 1
        'Revisa que las columnas de cuentas este llenadas correctamente
        'Verifica que la cuenta contable este llenada
        If Len(Trim$(SSDBGrid.Columns("Cuenta").CellValue(SSDBGrid.GetBookmark(int_Cor
                MsgBox Title:=gridCaption, _
                    Prompt:="El movimiento no tiene cuenta contable.", _
                    Buttons:=vbOKOnly + vbExclamation
                SSDBGrid.Bookmark = SSDBGrid.GetBookmark(int_Cont)
                SSDBGrid.RemoveItem (SSDBGrid.Row)
                Exit Function
        End If
        'Si estan vacios los cargos los pone en cero
        If IsEmpty(SSDBGrid.Columns("Cargo").CellValue(SSDBGrid.GetBookmark(int_Cont))
            SSDBGrid.Bookmark = SSDBGrid.GetBookmark(int_Cont)
            SSDBGrid.Columns("Cargo").Value = 0
        End If
```

Figura 9.20

PASO 6. CREACIÓN DE LA INTERFAZ DE USUARIO (CAPA DE PRESENTACIÓN)

Los detalles específicos en involucrados en la creación de la interfaz de usuario para una aplicación Windows DNA varían indudablemente de aplicación a aplicación, dependiendo del tipo de interfaz y tecnologías que se deseen emplear. Para nuestro caso práctico hemos decidido generar la interfaz de usuario (Capa de Presentación) sobre HTML y VbScript.

Para el desarrollo hemos dividido el proceso en 2 partes fundamentales:

- ✓ Diseño de cada página
- ✓ Proporcionar la interacción controlada entre los objetos emisarios y los diferentes objetos de la página

DISEÑO DE CADA PÁGINA

Hoy en día, los usuarios interactúan normalmente con las aplicaciones utilizando una variedad de elementos visuales, incluyendo botones, cajas de texto, listas, etc., que forman parte de una forma. Las aplicaciones están constituidas por varias formás y el proceso de diseñarlas implica adicionar y posicionar los diferentes elementos visuales

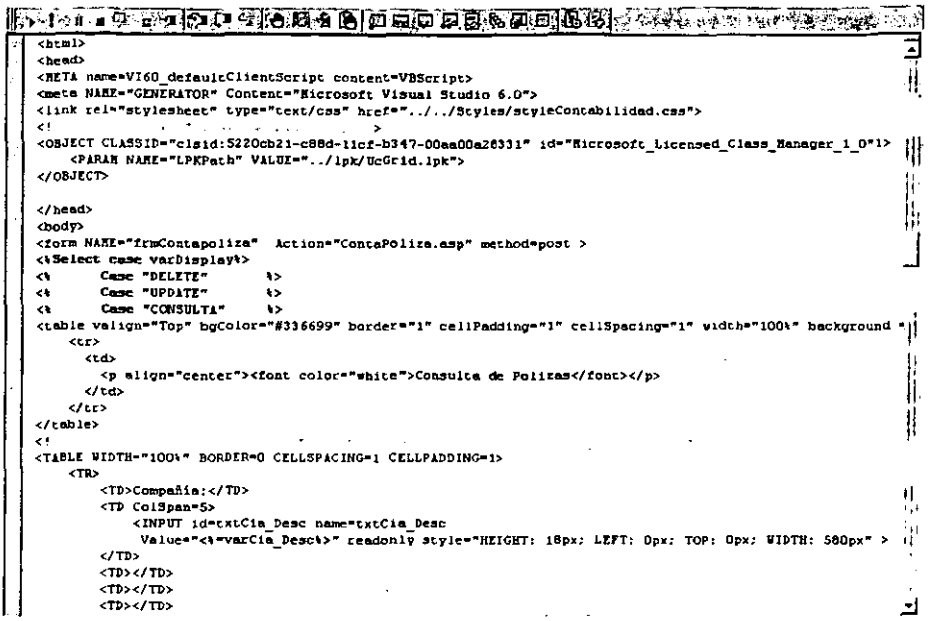
con los cuales interactúan los usuarios. Debido a que la mayoría de los usuarios ven a la aplicación en su totalidad como la interfaz con la que tienen contacto, el diseño de la interfaz de usuario es muy importante.

Para el diseño de la página de ingreso de pólizas se emplearon los siguientes elementos:

Tabla 9.32 Configuración de la forma de ingreso de Pólizas

OBJETO	TIPO
txtCia_Id	INPUT
cboReg_Id	LIST
cboFra_Id	LIST
CtlFecha	Activex Control
txtPeriodo_Id	INPUT
CboTipoPoliza_Id	LIST
txtContaPol_Concepto	INPUT
ctlarr_Detalle	Activex Control
txtSysUser_Nombre	INPUT
TxtTotalDebe	INPUT
TxtTotalHaber	INPUT
BtnInsertar	Imagen (Botón)

Con la ayuda de un editor de HTML necesitamos codificar la página utilizando los elementos antes mencionados y colocarlos cuidando que la estética de la página sea la más agradable posible y tomando muy en cuenta los aspectos de funcionalidad como son el orden de tabulación y el tipo de datos que se deben permitir ingresar en la página. El código completo de la página se encuentra en el apéndice C "Listado del Caso Práctico"



```
<html>
<head>
<META name=VI60_defaultClientScript content=VBScript>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<link rel="stylesheet" type="text/css" href="../../Styles/styleContabilidad.css">
<!                                                    >
<OBJECT CLASSID="clsid:5220cb21-c88d-11cf-b347-00aa00a28331" id="Microsoft_Licensed_Class_Manager_1_0"1>
    <PARAM NAME="LPKPath" VALUE="../lpk/UcGrid.lpk">
</OBJECT>

</head>
<body>
<form NAME="frmContapoliza"  Action="ContaPoliza.asp" method=post >
<%Select case varDisplay%>
<%      Case "DELETE"       %>
<%      Case "UPDATE"       %>
<%      Case "CONSULTA"     %>
<table valign="Top" bgColor="#336699" border="1" cellPadding="1" cellSpacing="1" width="100%" background "
    <tr>
      <td>
        <p align="center"><font color="white">Consulta de Polizas</font></p>
      </td>
    </tr>
</table>
<!
<TABLE WIDTH="100%" BORDER=0 CELLSPACING=1 CELLPADDING=1>
    <TR>
        <TD>Compañía:</TD>
        <TD ColSpan=5>
            <INPUT id=txtCia_Desc name=txtCia_Desc
             Value="<%=varCia_Desc%>" readonly style="HEIGHT: 18px; LEFT: 0px; TOP: 0px; WIDTH: 580px" >
        </TD>
        <TD></TD>
        <TD></TD>
        <TD></TD>
```

Figura 9.21

Proporcionar la interacción controlada entre los objetos emisarios y los diferentes objetos de la página

Una vez que la parte estética de la página ha sido diseñada, es turno de dar funcionalidad a la página, es decir tenemos que proporcionar la interacción controlada entre los objetos emisarios y los elementos de la página. Esto lo hacemos codificando en la página las peticiones a operaciones de negocios, o lo que es lo mismo, instanciando los métodos de nuestros objetos emisarios de la capa de más arriba. Para nuestro caso, hacemos uso del componente bus_Contabilidad, util_Contabilidad y de sus diferentes interfaces según sea la operación que deseamos realizar.

La siguiente imagen muestra parte de la codificación del ingreso de pólizas.

```
else

    select case varAction
    case "INSERTAR"
        On error Resume Next
        'Obtiene los valores enviados por el usuario
        varFra_Id               =Request.Form("cboFra_Id")
        varTipoPoliza_Id        =Request.Form("cboTipoPoliza_Id")
        varContaPol_FechaAlta   =Request.Form("ctlFecha")
        varContaPol_Concepto    =UCase(Request.Form("txtContaPol_Concepto"))
        vararr_Detalle          =Request.Form("hdnarr_Detalle")


        'Obtiene el arreglo a partir de la cadena
        set objUtilerias=Server.CreateObject("Util_Contabilidad.Utilerias")
        arr_Detalle     =objUtilerias.GeneraArreglo(vararr_Detalle)
        if Err.number  <> 0 then
            Set objUtilerias = nothing
            Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  & "&URL=./Polizas/C
        end if
        Set objUtilerias = nothing

        'Con los valores obtenidos por el usuario
        Set objInsert= Server.CreateObject("bus_Contabilidad.Insert")
            varContaPol_Id = objInsert.Poliza varSysUser_Id,varCia_Id,varReg_Id,varFra_Id,varTipoPoliza_Id
                        varContaPol_Concepto,varContaPol_FechaAlta,arr_Detalle,"TRUE"
        if Err.number  <> 0 then
            Set objInsert = nothing
            Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  & "&URL=./Polizas/C
        end if
        set objInsert= Nothing
        Response.Redirect "ContaPoliza.asp?Cia_Id=" & varCia_Id & "&Reg_Id=" & varReg_Id & _
                          "&Fra_Id=" & varFra_Id & "&TipoPoliza_Id=" & varTipoPoliza_Id & _
                          "&Periodo_Id=" & varPeriodo_Id & "&ContaPol_Id=" & varContaPol_Id
    case "BORRAR"
```

Figura 9.22

Una vez terminada la página tiene la siguiente apariencia:

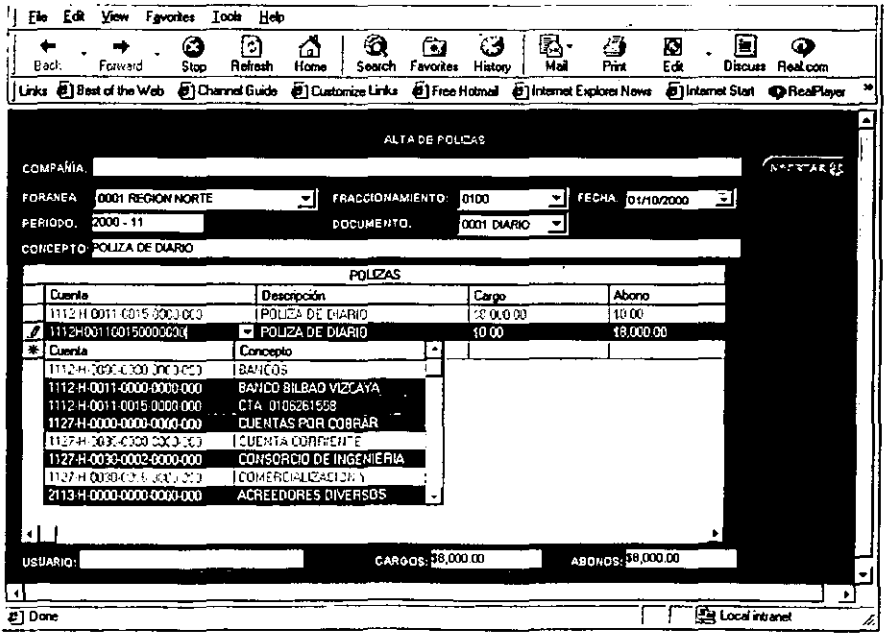


Figura 9.23

PASO 7. IMPLEMENTACIÓN Y CONFIGURACIÓN DEL SISTEMA

Antes de que nuestra aplicación pueda ser implementada, se deben configurar los diferentes componentes del sistema para que interactúen entre sí. La configuración de el servidor de base de datos que propiamente después de la instalación, a no ser que se tenga la necesidad de configurar el conjunto de caracteres a utilizar o la precisión de los tipos de datos que se desplieguen, para nuestro caso, esto no fue necesario.

El mismo caso se da con el servidor Web que fue propiamente instalado en el paso uno[6].

Un aspecto que no podemos dejar de mostrar es la configuración de los objetos de negocios. Para configurar el funcionamiento de los objetos de negocios dentro del broker Transaction Server:

1. Inicie el Transaction Server Explorer
2. Para crear un nuevo paquete de clic derecho en el nodo de Packages Installed y seleccione Package del menú New.

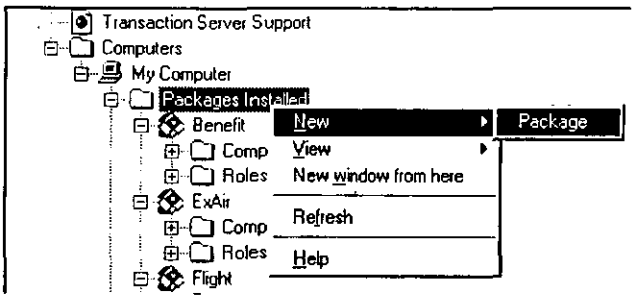


Figura 9.24

3.Del Package Wizard ...
Seleccione Create an empty package

[6] Configuración de los sistemas de hardware y software requeridos

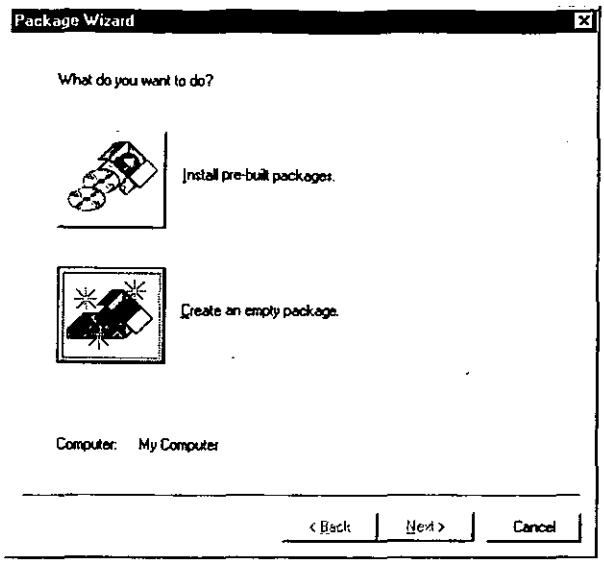


Figura 9.25

4. Proporcione un nombre para el paquete
5. De clic en Finish

Una vez creado el paquete, el siguiente paso es adicionar los objetos de negocios al paquete.

6. Para adicionar un componente al paquete, de clic derecho en el nodo del paquete y seleccione Componet del menú New.

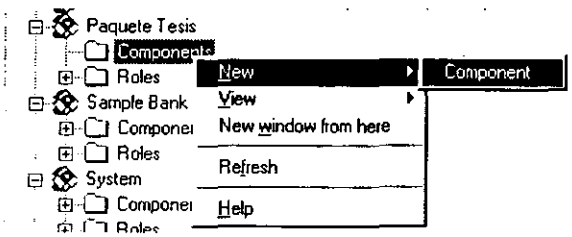


Figura 9.26

6. Del Component Wizard ...
   Seleccione Install new component(s)

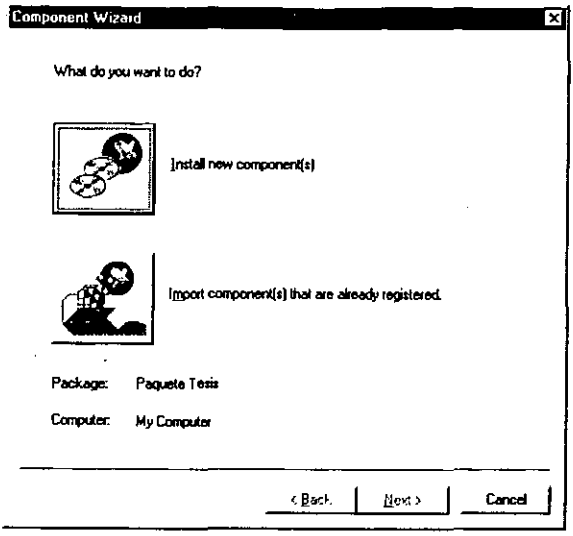


Figura 9.27

7. En la pantalla de Install Components ...
   Proporcione la ruta de los tres componentes que diseñamos, estos son:
   Bud_Contabilidad.dll
   Dat_Contabilidad.dll
   Util_Contabilidad.dll

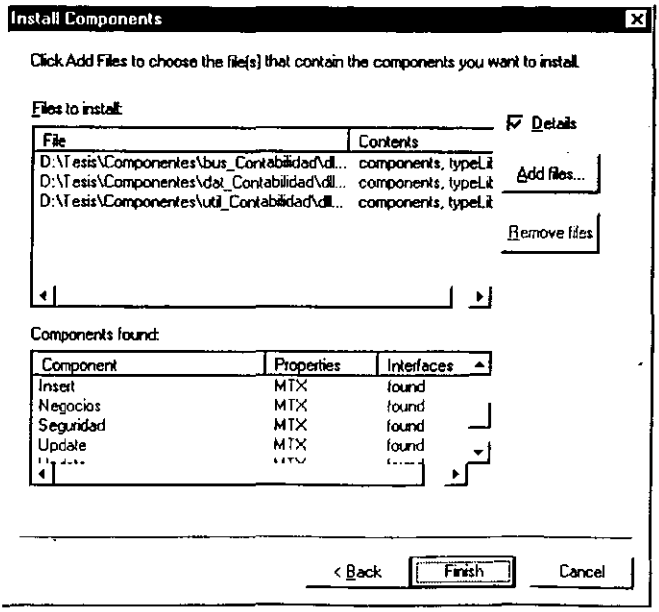


Figura 9.28

8. Una vez instalados los componentes dentro del broker ven de la siguiente forma:

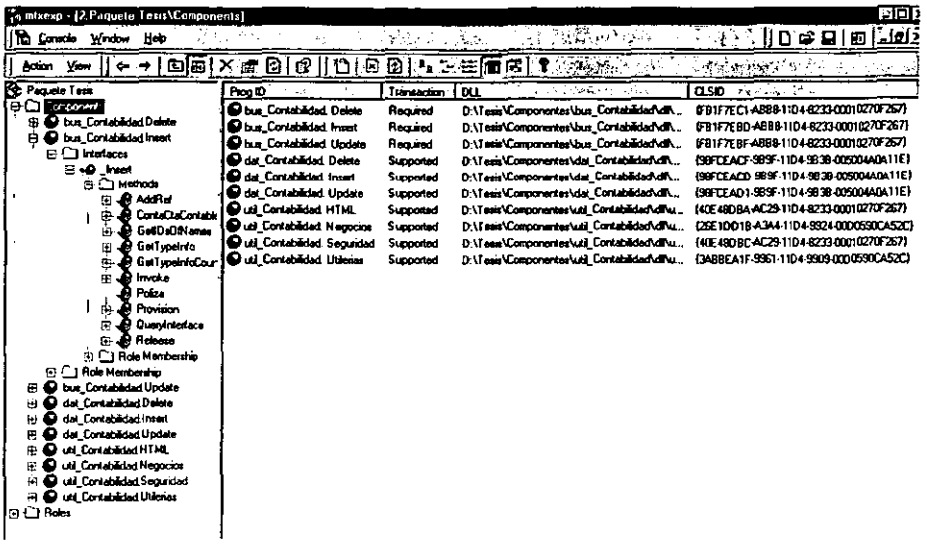


Figura 9.29

Por último cabe hacer mención que al instalar los componentes dentro del broker, en ningún momento definimos el tipo de transacción que soportan estos componentes, y es que esto lo hicimos durante el diseño de los componentes con la propiedad MTSTransactionMode, no obstante es posible, una vez instalados los componentes dentro del broker, definir el tipo de transacción que soportarán.

Con esto terminamos el caso práctico y los pasos siguientes dentro de la implementación, tienen que ver con los cotidianos para cualquier sistema, como son la instalación de los clientes, que en nuestro caso no representa mayor problema ya que solo es necesario contar con un explorador de Web. Esto definitivamente es una de las muchas ventajas que tiene el desarrollo de aplicaciones distribuidas bajo el esquema de Windows DNA.

CONCLUSIONES

La tesis *"Windows DNA en el desarrollo de un Sistema Nervioso Digital"*, que hemos presentado, tiene el propósito primordial de dar a conocer una metodología de trabajo que permite el desarrollo de aplicaciones de forma eficiente, tomando las mejores características de los antiguos modelos de desarrollo de aplicaciones, enfocando estas características y haciendo uso de las nuevas tecnologías como las mencionadas durante el desarrollo de la tesis y de la misma Internet.

El propósito de este trabajo se enfocó en presentar y dar un ejemplo palpable y funcional del desarrollo de un sistema de nivel empresarial y de misión crítica (Un Sistema Nervioso Digital) haciendo uso de la arquitectura de Windows DNA, con lo que demostramos que el desarrollo de aplicaciones de misión critica no esta supeditado a grandes servidores y un conjunto de software costoso que requiere de personal altamente calificado y certificado en la materia. El esquema que presentamos define el desarrollo de una aplicación empresarial haciendo uso de herramientas comerciales y al alcance de las pequeñas y medianas empresas.

En México existen, sin lugar a duda existe un gran número de empresas grandes, para las cuales no es problema el invertir en tecnología para poder organizar su flujo de información y estar a la vanguardia tecnológica, manteniéndose siempre competitivas. Pero, lo que es una verdad irrefutable, es que también existen pequeñas y medianas empresas que, debido a su crecimiento en operaciones, necesitan también el organizar su flujo de información, pero que aún no cuentan con el capital necesario para hacer una fuerte inversión tecnológica, y lo más preocupante es que el número de estas empresas sobrepasa por mucho al conjunto de las grandes empresas. Entonces, ¿Qué es lo que estas empresas pueden hacer para no encontrarse en desventaja con las grandes empresas? ¿Qué hacer cuando no se tienen miles o incluso millones de dólares para la adquisición e implantación de un software para la automatización de procesos y la organización de los flujos de información dentro de la empresa?

Creemos firmemente que la propuesta en nuestra tesis beneficia a todas aquellas empresas que no pueden soportar la compra de software costoso y muchos menos el cambio de una plataforma de hardware.

Ahora bien, este esquema de n-capas y aplicaciones distribuidas tiene otras grandes ventajas, además de para los pequeños y medianos empresarios, para los desarrolladores e implantadores de tecnologías. Estas ventajas van desde la reducción en el tiempo de desarrollo hasta el costo en la implementación de la aplicación, aboliendo todas las desventajas que tiene el modelo Cliente/Servidor y, porque no decirlo, tomando muchas de las ventajas del mismo.

Entre las ventajas más significativas se encuentran:

PARA LOS PROFESIONALES DE IT

- La carga de trabajo se puede distribuir en varios servidores e incluso dejar parte en el cliente.

- El control de versiones prácticamente desaparece, ya que solo existe una sola versión de la aplicación.

- Una vez terminado el desarrollo, la implementación se hace en muy poco tiempo y los cambios a la aplicación se ven reflejados inmediatamente en los clientes.

- Las reglas de negocio se encuentran centralizadas, de manera que un cambio en el negocio se puede implementar fácilmente.

- La escalabilidad de la aplicación prácticamente depende de los recursos de hardware con que se cuenten, de manera que la aplicación puede crecer con las necesidades de la empresa.

- El desarrollo de la aplicación se puede distribuir fácilmente debido a que se trabaja con interfaces entre las diferentes capas, se pueden tener avances paralelos en el desarrollo.

- No existe la necesidad de instalar la aplicación en el cliente, ya que el cliente al tener acceso al servidor Web, recibe una imagen funcional de la aplicación.

- El manejo de las operaciones es transaccional y automático.

- Se pueden implementar diferentes niveles de seguridad dentro de la aplicación, además de los esquemas tradicionales de seguridad.

PARA LOS DUEÑOS DE LAS PEQUEÑAS Y MEDIANAS EMPRESAS

- El costo de mantenimiento se reduce considerablemente, debido a que la aplicación esta centralizada, desaparecen los costos de viáticos y costos relacionados con que un profesional de IT tenga que abandonar las oficinas centrales para dar mantenimiento a un cliente que se encuentre geográficamente distanciado.

- El costo de desarrollo depende de la plantilla de desarrolladores con que se cuente y de la complejidad de funcionamiento de la empresa. Para una empresa tipo que requiere de los sistemas básicos para funcionar como con la Contabilidad, Cuentas por Pagar, Cuentas por Cobrar, Almacén y Nomina, hablamos de 5 módulos. Para nuestro caso práctico se realizó un módulo de Contabilidad que se desarrollo a cabo en mes y medio, una vez hecho el análisis. En el mercado a finales del año 2000, el promedio de sueldos para desarrolladores es el siguiente:

TIPO	SUELDO
Lider de Proyecto	Desde $20,000.00 hasta $25,000.00
Programador	Desde $15,000.00 hasta $18,000.00

Con lo anterior podemos entonces decir que el costo de desarrollo e implementación es uno solo y que sigue siendo mucho más accesible que la adquisición de una solución de miles o millones de dólares, que tal vez resulte sobrada para las empresas en cuestión.

- No es necesario invertir en una nueva plataforma de cómputo, si es que se cuenta con una red computadoras personales.

- El costo por licencias es mínimo ya que la aplicación se encuentra centralizada y los clientes pueden ser cualquier browser, que actualmente se encuentran sin costo en la Internet.

- La inversión en infraestructura de líneas telefónicas dedicadas o de redes virtuales privadas, no es necesaria ya que se puede hacer uso de la Internet implementando un buen esquema de seguridad.

Como hemos presentado, las ventajas son grandes y muy interesantes ya que resuelven la mayoría de los inconvenientes que pudiese tener una aplicación bajo el esquema tradicional Cliente/Servidor de dos capas o de dos y media capas.

Durante el desarrollo de la tesis nos ayudamos mucho de las herramientas Microsoft y de hecho nuestro caso práctico fue realizado con este tipo de herramientas. Además, aunque la propuesta de un Sistema Nervioso Digital se le atribuye a Microsoft, si queremos hacer notar y dejar bien establecido que el trabajo que hemos presentado no es por ningún motivo una tesis encaminada al uso exclusivo de las herramientas de Microsoft.

En cada uno de los capítulos tratamos de ofrecer la parte análoga de la tecnología con otras firmas y plataformas como lo son Oracle, SUN Microsystems, los diferentes sabores de Unix, Netscape, etc. Si debemos hacer énfasis que nuestro trabajo presenta una metodología de trabajo y hace uso de una arquitectura tecnológica y que no depende de una plataforma de mercado específica, así que la solución puede encajar perfectamente en cualquier plataforma o conjunto de plataformas tecnológicas. Así para la capa de datos se puede hacer uso a conveniencia de cualquier manejador de base de datos como Oracle, DB2, Sybase o Informix. Para la capa de negocios se puede hacer uso de CORBA, Enterprise Java Beans o una mezcla de estos con MTS y por último para la capa de presentación se pueden utilizar cualquier número de herramientas existentes para la generación de las páginas como por ejemplo, JavaScript, HTML, Cold Fussion, PHP, Java Applets, etc.

La razón por la que nosotros optamos por la plataforma Microsoft fue por mera conveniencia en cuanto a la disponibilidad de las herramientas y el sistema operativo.

Es muy importante mencionar que el desarrollo de esta tesis es nuestra propuesta para solucionar una eminente necesidad que se da en las empresas mexicanas por sistemas de misión critica que puedan satisfacer sus necesidades de funcionamiento. En México existen empresas que no cuentan con una infraestructura tecnológica o que inclusive en algunos casos no cuentan con una cultura informática, pero que el advenimiento de la globalización les esta exigiendo ser competitivas, lo que se traduce en hacer más efectivos sus procesos y estar preparadas para los cambios que hoy en día se presentan cada vez con más frecuencia.

La globalización esta exigiendo que las empresas en todo el mundo sean competitivas y el no poder satisfacer esta necesidad, muy probablemente repercuta en la desaparición de aquellas que no puedan en un momento dado ofrecer un servicio eficiente a sus clientes en el mercado. La competencia ahora no es con las empresas en México nada

mas, sino con empresas en todo el mundo, así un retrazo en la entrega de un bien o un servicio puede ser la diferencia entre preferir a una empresa y no a otra. Creemos que esto ya se esta dando actualmente en México, al grado de tener que afrontar por ejemplo que una empresa extranjera dedicada a la venta de libros *"Amazon.com"*, sin estar establecida en México tenga más ingresos que otras empresas mexicanas del mismo ramo. Confiamos en que este trabajo pueda ayudar a resolver una problemática sobre la cual algunos tal vez aún no tienen noción pero que es inevitable y que en su momento se comprenderá.

APÉNDICE A

FUNDAMENTOS DE DISEÑO DEL MODELO DE PROGRAMACIÓN

Cuando un diseñador desarrolla un componente debe elaborar un modelo de programación. Cuando se codifica un componente bajo el Component Object Model (COM) debe decidir cómo es que el componente será programado, es decir, cómo se escribirá el código para manipular al componente.

Lo anterior nos conduce a una serie de preguntas: ¿Cuándo se debe usar una propiedad, un evento y cuándo un método?, ¿Cómo se debe nombrar a las propiedades o a los métodos?. A menudo no es posible encontrar respuesta a estas interrogantes, pero se puede deducir una serie de reglas para el diseño de componentes examinando los modelos de programación existentes que son de dominio publico. Aunque esto puede ser bastante arriesgado, sobre todo cuando los diseñadores de esos modelos no utilizan una lógica previamente bien definida.

Por Modelo de Programación se entiende al conjunto de interfaces, propiedades, métodos y eventos que expone un componente y que le permiten a un diseñador manipularlo para escribir los programas.

Diseñar un buen modelo de programación es tan importante como diseñar una buena interfaz de usuario (User Interface UI), muchos de los principios usados en el diseño de UI, se aplican directamente al diseño de un modelo de programación. Las buenas prácticas en del diseño de UI permiten al usuario trabajar en un nivel mayor de abstracción, por lo que UI presenta una vista lógica de la funcionalidad a diferencia de la realidad física de esa funcionalidad. Es decir, expresa cosas de una forma que se equipara a la forma en que el usuario piensa, la cual no necesariamente es como el sistema trabaja realmente.

Un buen modelo de programación no expone su estructura interna, expone sus funciones en un nivel más alto de abstracción de modo que el cliente (el diseñador) pueda concentrarse en lo que se desea hacer y no en cómo lograr una simple tarea.

MODELO DE PROGRAMACIÓN

El termino "Modelo de Programación" es mucho mas adecuado para describir lo qué intentamos diseñar. Así se debe fundamentar todo lo concerniente al desarrollo de los componentes, pues estos serán la base del desarrollo de otros niveles más altos de abstracción.

En la siguiente parte del capítulo se tratan algunas reglas básicas, que se han desprendido durante la experiencia en el desarrollo de numerosos modelos de programación, el seguimiento de estas y las teorías asociadas a ellas han conducido a exitosos y poderosos modelos de programación.

Nótese que estas reglas son simples guías, y algunas veces es recomendable romper con las reglas cuando tenga sentido hacerlo, lo importante es llevar a cabo las ideas básicas que estas encierran, si alguna de estas reglas no coincide con las ideas de nuestro modelo, podemos prescindir de dicha regla.

1. MANTENER LA CONSISTENCIA O UNIFORMIDAD.

La regla más importante de todas es mantener la consistencia o uniformidad. La consistencia se da a dos niveles. En el primer nivel la consistencia se debe presentar durante todo el desarrollo de un componente, ya que es frustrante para los programadores trabajar de una forma con el componente en una parte del modelo de programación y de otra totalmente distinta en otra parte. En el segundo nivel, la consistencia se debe dar de acuerdo con modelos de programación ya existentes y a los cuales ya se encuentren acostumbrados los usuarios de nuestros componentes. Esto no siempre es factible de seguir ya que en el mercado existen librerías que se contradicen completamente unas con otras en su forma de trabajo.

2. DISEÑO UTILIZANDO ESCENARIOS

Si la regla mas importante es mantener la consistencia, el diseño de los modelos de programación en base a escenarios, es la segunda en importancia, y al mismo tiempo es la regla que mas veces se ignora.

Cuando se diseña una interfaz de usuario, pensamos sobre lo que comúnmente haría el usuario dentro de la interfaz y hacemos lo posible para que esas tareas resulten lo más sencillas posible. De la misma manera, los modelos de programación deben ser diseñados con escenarios comunes en mente, respondiendo a preguntas tales como: ¿Qué es lo que comúnmente haría un programador con el componente?, ¿Qué características son para usuarios más avanzados y cuáles son las mas fáciles de usar?. Nótese que se hace mucho énfasis en el resultado final que ofrece el componente y no en sus objetos, estructuras de datos o funciones internas.

Por ejemplo, supongamos que nos encontramos diseñando un modelo de programación para un componente que comprenderá la funcionalidad del protocolo de transmisión de archivos (FTP); para la implementación de este componente se debe trabajar con elementos como servidores, *sockets*, encabezados, cadenas, directorios, archivos, etc. Pero todos estos elementos no son los que los desarrolladores que utilicen este componente desean manejar, sino la funcionalidad que estos les pueden proporcionar.

De esta manera, una vez terminado el componente, puede ser codificado por algún programador de la siguiente forma:

```
Set objInsert = CreateInstance("dat_Contabilidad.Insert")
ObjInsert.Poliza Cia_id,Reg_id,Fra_Id,TipoPoliza_Id, _
       Periodo_Id,arr_Detalle
Set objInsert = Nothing
```

Tómese en cuenta que toda esa funcionalidad avanzada, como lo es el manejo explicito de la conexión, el copiado asíncrono de archivos con notificación de progreso, etc., que le permiten al desarrollador obtener un archivo de una base de datos o de un servidor de correo, aun se encuentra disponible para ser usada. El punto es que el desarrollador no necesariamente tiene que hacer uso de ese manejo complejo, que también ofrece el componente, cuando solo necesite realizar tareas sencillas.

Podemos decir, entonces, que el principio del diseño es: "Hacer fácil las tareas comunes y hacer posible las tareas complicadas". Un ejemplo de la vida real lo

podemos encontrar en las videocaseteras. Casi cualquier persona puede introducir un cassette en estas y observar una película de una forma muy simple; esa persona no necesariamente tiene que saber que la vidoecaseteras pueden ser programadas para 50 eventos diferentes: congelar la imagen y avanzar cuadro a cuadro, recorrer la cinta hacia delante o hacia atrás, etc. Todas esas características de la videocasetera se encuentran ahí para ser usadas, pero no se encuentran a la mano de un usuario inexperto.

3. CUÁNDO UTILIZAR PROPIEDADES Y CUÁNDO METODOS

Para determinar si algo es una propiedad o un método, podemos utilizar las siguientes reglas:

- ✓ Todo aquello que refleje el estado de un objeto, debe ser expuesto como una propiedad. Los ejemplos son Caption, Enabled, Visible, etc.

- ✓ Si es un objeto cuyo estado es de solo lectura (read-only), también debe ser expuesto como una propiedad pero debe ser de solo lectura (esto es, no incluye un procedimiento Set/Let). Como ejemplo tenemos hWnd, hDC, o WindowStyle.

- ✓ Si obtener un valor sobre un objeto no tiene un efecto especial sobre este, entonces estamos hablando de una propiedad y no de un método. Un procedimiento del tipo Get, en Visual Basic, puede involucrar el extraer un valor de una base de datos la primera vez que es llamado, pero esas operaciones no las podrá notar el desarrollador que utilice la interfaz.

- ✓ El obtener el valor de una propiedad no debe depender de darle u obtener el valor a otra. No debe haber diferencia si primero se obtiene el valor de la propiedad A y luego de la propiedad B o viceversa.

- ✓ Cualquier operación que lleve a cabo una acción y que no tenga que ver con la obtención de un valor, debe ser un método. Como ejemplos tenemos Open, Save, Export, Add, Remove, etc.

Existen pocas reglas, pero no por esto menos importantes, para el nombramiento de propiedades y métodos:

- ✓ Los nombres de las propiedades generalmente son sustantivos o adjetivos (Por ejemplo Caption, BackColor, ConnectionString, etc.)

- ✓ Los nombres de los métodos por lo general son verbos (Por ejemplo Save, Connect, Randomize, etc.)

- ✓ Se debe evitar el uso de nombres negativos, especialmente para las propiedades booleanas. Esto resulta en una sentencia doblemente negativa en el código cuando los programadores igualen esta propiedad a Falso, lo que posteriormente resulta muy difícil de entender (Por ejemplo `objNegocios.NoRecalcula=False`)

4. NOMBRES DE PARÁMETROS, TIPOS DE DATOS Y DEFAULTS

Al diseñar métodos para las interfaces de programación, también se deben considerar los nombres, tipos de datos y defaults para los parámetros de esos métodos. Aquí también existen algunas reglas básicas que se deben de seguir al diseñar los parámetros de los métodos:

Aunque los desarrolladores en C/C++ son propensos a utilizar la notación Húngara (donde los nombres de los parámetros se encuentran precedidos por un conjunto de caracteres que representa su tipo de dato o la función de ese parámetro), normalmente se debe evitar esta práctica. La notación Húngara casi no es usada en componentes ya existentes y los desarrolladores de VB pueden encontrar esto un tanto confuso.

El nombre del parámetro debe describir que información es requerida, de tal forma que el desarrollador no tenga que indagar que dato pasar. Los nombres de los parámetros deben ser tipo titulo y sin espacios (Ejemplo: ConnectionString)

Los parámetros deben ser de un tipo de dato específico siempre que esto sea posible, y se puede hacer uso del tipo de dato variant para aquel caso en que sea estrictamente necesario aceptar diferentes tipos de datos. Esto conlleva a tener menos errores en el código de los programadores que hacen uso del componente, y define qué es lo que el parámetro requiere.

Se deben utilizar enumeraciones si el parámetro puede tomar un conjunto fijo valores. Los editores de código como el de VB, VBA y otros, ofrecen al desarrollador una lista de opciones válidas que le permiten al desarrollador completar su código reduciendo la posibilidad de errores.

Si el parámetro es una bandera booleana, entonces debe ser declarada como tal y no como *Long* o *DWORD*. Por ejemplo, en *Automation* el valor de *True* es igual a –1 y quien no tome esto en cuenta podría esperar un valor de 1 en lugar de un *True* si el parámetro ha sido declarado como *Long*.

Los parámetros deben ser declarados como opcionales solo si en realidad estos no son requeridos, si existe un default sobre estos o si se les puede dar valor por medio de una propiedad.

Todos los parámetros opcionales deben tener valores por default, así también deben ir a la derecha de los parámetros requeridos aunque esto no sea explícitamente requerido por Automation. Esto repercute en hacer nuestro modelo de programación fácil de entender. Los parámetros deben ser ordenados en forma lógica de tal forma que sea intuitivo su uso para el desarrollador.

Se debe evitar el uso de literales o números "mágicos" para los valores por default o los valores con un significado especial. Por ejemplo en la propiedad AbsolutePosition en ADO existe una enumeración que reemplaza valores de BOF/EOF, de tal forma que en lugar de preguntar por un número mágico como –1 el desarrollador puede codificar de la siguiente manera:

`If adoRS.AbsolutePosition=adPosUnknown`, que es mucho más fácil de leer y entender.

Seguir estas reglas durante el diseño de los parámetros hace a nuestro modelo de programación más entendible.

5. EVENTOS

Los eventos, definitivamente, son un poco mas complejos que las propiedades y los métodos. En general los eventos son usados para notificar al cliente que algo ha sucedido y de esta manera se da la oportunidad de ejecutar alguna acción en respuesta. Es muy común encontrar que los controles utilizan los eventos de forma imprescindible, debido a que tienen la necesidad de darle a conocer al desarrollador que el control ha recibido – por ejemplo -- un clic o un doble clic. De la misma forma que los controles, los componentes pueden disparar eventos y permitir la creación de programas muy poderosos que hagan uso de ellos.

Una de las principales cosas a realizar cuando se diseñan eventos es definir el comportamiento de lo que se denomina *"reentrancia"*, es decir se debe definir que sucede cuando un desarrollador escribe código que manipula al componente durante un evento que es disparado por el mismo. Imagine un control que dispara al evento *Change*, y durante el evento *Change*, el desarrollador escribe código que cambia el contenido del control nuevamente. ¿Debe el evento *Change* dispararse nuevamente?

La capacidad para manejar la *reentrancia* puede variar desde permitir casi todo hasta permitir casi nada. Normalmente los componentes deben evitar que otros eventos sean disparados durante la ejecución de un evento. En el caso de que se decida permitir que otros eventos sean disparados durante la ejecución de otro evento, se debe tener cuidado de que un programador pueda inocentemente provocar un ciclo recursivo.

6. DOCUMENTACIÓN

Finalmente el desarrollo de un componente requiere de una buena documentación que puede incluir ayuda implícita en forma de ToolTips, Status Bars, etc., en las aplicaciones de tal forma que evite al usuario tener que ir en busca del archivo de ayuda y desviarse de la tarea que se encuentre realizando. De la misma forma es posible proporcionar ayuda importante de forma rápida, la cual forma parte del modelo de programación, de tal manera que el desarrollador que use la propiedad, el método, el evento o la clase, sepa el significado de cada uno de estos sin tener que recurrir a la ayuda. Para poder llevar a cabo lo anterior, es necesario introducir cadenas de documentación en la librería de tipos del componente. Estas cadenas son mostradas en la mayoría de las herramientas de desarrollo en los "object browser" y en algunas ocasiones en los editores de código.

Estas cadenas de documentación pueden no ser relevantes para los creadores de los componentes, pero para los desarrolladores que hacen uso de estos son de gran ayuda, ya que incluyen descripciones con significados exactos sobre la interfaz de programación. Por ultimo es conveniente mencionar que al diseñar un modelo de programación se deben incluir estas cadenas de documentación sobre propiedades, métodos, eventos e interfaces.

APÉNDICE B

ARQUITECTURA LÓGICA ESCALABLE BASADA EN COMPONENTES (COMPONENT-BASED SCALABLE LOGICAL ARCHITECTURE CSLA)

Para utilizar componentes y objetos de negocios de forma efectiva en el desarrollo de aplicaciones es necesario el uso de una buena arquitectura lógica, lo anterior significa tener una firme idea de los tipos de servicios que proporcionara nuestra aplicación y si estos servicios se implementarán en la interfaz de usuario, en los componentes o en el servidor de bases de datos.

La CSLA esta diseñada para trabajar en una sola estación de trabajo o bien en red, permitiendo de esta forma poder escalar una aplicación desde una sola computadora hasta una red completa con un esfuerzo mínimo.

Las premisas de la CSLA son:

- ✓ Facilidad para la implementación física a partir de modelos lógicos flexibles.
- ✓ Permitir la reutilización del código a través de un modelo de diseño orientado a componentes.
- ✓ Proporcionar una arquitectura escalable, soportando aplicaciones sobre una máquina o en cientos de máquinas en red.

Un entendimiento preciso de los conceptos lógicos detrás de la arquitectura física es critico para el uso efectivo de la CSLA.

ARQUITECTURA LÓGICA

Es muy importante recordar que la CSLA es una arquitectura lógica. La CSLA por si misma no especifica cuantas máquinas estarán corriendo porciones de la aplicación, ni especifica exactamente cuanto procesamiento se llevará a cabo en determinada máquina.

Como modelo lógico, la CSLA ofrece una guía general que se puede utilizar al determinara la arquitectura física mas óptima que se puede usar.

Anteriormente se ha mencionado la arquitectura Cliente/Servidor de tres capas donde la aplicación se encuentra dividida de la siguiente forma:

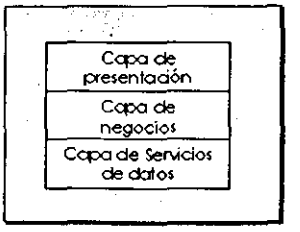


Figura B.1

Aunque este modelo es poderoso, no es lo suficientemente flexible. En particular, con este modelo estamos colocando físicamente la capa de negocios ya sea en el cliente o en el servidor. Esto puede ser muy poco óptimo en muchos casos, principalmente porque no toda la lógica de negocios es siempre la misma; alguna lógica de negocios requiere de mucho acceso a datos, otra lógica de negocios no requiere el acceso a los datos tan frecuentemente, pero si es utilizada para crear una interfaz de usuario robusta para la validación de la entrada de datos u otras interacciones con el usuario.

Si se requiere que el usuario experimente una interfaz rica y robusta se necesita utilizar validación a nivel de campo, validación a nivel de pantalla y tal vez algunos cálculos en tiempo real u otra lógica de negocios. Podemos considerar este tipo de lógica de negocios como lógica central de interfaz de usuario (UI-centric logic), dado que es usada por la interfaz de usuario. Con los modelos tradicionales de Cliente/Servidor tenemos la opción de ubicar esta lógica físicamente del lado del cliente o bien centralizarla en el servidor y replicar el código en la interfaz de usuario, para proporcionar una interfaz de usuario de alta calidad. Ninguna de estas opciones es lo suficientemente atractiva. Poner toda la lógica de negocios de lado del cliente puede hacer que el acceso a los datos se vuelva lento y como consecuencia reduzca el performance. Poner la lógica de negocios del lado del servidor y replicar la lógica a la interfaz, implica un costo de mantenimiento muy alto. La solución es escribir la lógica de negocios tanto en la interfaz de usuario como en el servidor, proporcionando gran apertura e incremento de la efectividad en el desarrollo.

Lo que CSLA hace, desde un perspectiva lógica es construir un modelo Cliente/Servidor, incorporando el concepto de lógica de negocios UI-Centric y Data-Centric. Al dividir el concepto de "Capa de Negocios" en dos capas, proporciona mas flexibilidad al diseño de las aplicaciones.

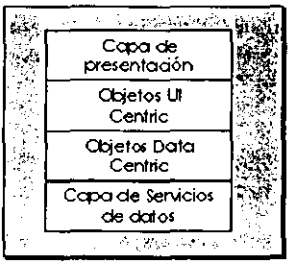


Figura B.2

Podemos ubicar físicamente la lógica de negocios UI-Centric y Data-Centric en varias computadoras para optimizar el performance y poder ofrecer una interfaz de usuario rica y robusta.

LA CSLA ESTÁ BASADA EN COMPONENTES

La CSLA esta basada en componentes (Component-based), por lo que se puede asumir que una aplicación diseñada sobre la CSLA esta conformada de varios componentes que trabajan en conjunto para proporcionar la funcionalidad deseada. Dado que es

component-based la aplicación puede interactuar con los componentes creados o proporcionados por otras fuentes.

En cierto sentido, se puede decir que cada una de las cuatro capas lógicas que conforma la aplicación esta compuesta de uno o mas componentes. Esto significa que tenemos la interfaz de usuario (UI), la lógica de negocios UI-Centric, la lógica de negocios Data-Centric y el procesamiento de datos como un conjunto de componentes que interactúan unos con otros.

Este es un modelo poderoso, por ejemplo, considérese a la UI como un conjunto de componentes que podemos reemplazar fácilmente con otro conjunto con poco o casi nada de impacto sobre las otras tres capas de la aplicación.

LA CSLA ES ESCALABLE

La CSLA es inherentemente escalable, debido principalmente a su flexibilidad. La escalabilidad no viene solo de un modelo lógico, sino también de cómo se aplique el modelo lógico a la arquitectura física. Dado que la CSLA es suficientemente flexible, se cuenta con varias opciones al implementar las diferentes capas lógicas de la aplicación hacia el ambiente físico otorgándonos así el poder de crear diseños altamente escalables.

En un escenario determinado, si las estaciones de trabajo cliente tienen gran poder de procesamiento y también la red tiene un gran ancho de banda se puede optar por asignar una gran cantidad de procesamiento del lado del cliente para tomar ventaja del poder que otorga el procesamiento distribuido.

La razón por la cual, en este caso, es importante un gran ancho de banda se debe a que el cliente puede hacer peticiones de grandes cantidades de datos y puede ser que muchos de los datos solicitados no sean indispensables, pero eso se puede saber solamente después de que el cliente haya procesado esos datos.

En un escenario mas típico, las estaciones de trabajo cliente pueden tener un gran poder de procesamiento, pero el ancho de banda de la red se puede encontrar limitado. La CSLA nos permite poner la UI y una cantidad mínima de procesamiento del lado del cliente, para tomar ventaja del computo distribuido, y de la misma forma nos permite poner la lógica y el procesamiento de datos en servidores centralizados para minimizar el tráfico en la red.

En este caso se puede minimizar lo que es enviado a través de la red aplicando lógica de negocios en el servidor para aislar y distinguir los datos significativos. Una vez que el servidor ha reducido los datos a solo los esenciales, pueden enviarse por la red, minimizando así el tráfico.

En otro escenario, las estaciones de trabajo cliente pueden encontrarse limitadas, o bien se requiere minimizar al máximo los componentes implementados en el cliente. En tal caso se puede poner solamente la UI en el cliente, con todo el procesamiento y la lógica de negocios en los servidores centralizados. Con este escenario se ofrece una interfaz de usuario muy limitada, pero obtenemos un alto rendimiento y escalabilidad.

EL MODELO FÍSICO

Dado que escalabilidad y performance se encuentran íntimamente ligados no sólo al modelo lógico de la CSLA, sino también a la implementación física, a continuación revisaremos las arquitecturas físicas mas importantes para nuestro estudio y veremos como es que la CSLA aplica para cada una de estas.

De las arquitecturas físicas, tres destacan por su importancia:

- ✓ Arquitectura de n-capas (n-tier)
- ✓ Arquitectura de una sola computadora (single computer)
- ✓ Arquitectura basada en browser (browser-based ó cliente inteligente)

ARQUITECTURA DE N-CAPAS (N-TIER)

La arquitectura física (n-tier) es la que comúnmente encontraremos al crear aplicaciones empresariales. Tales aplicaciones demandan una confiabilidad y escalabilidad muy altas y están diseñadas de tal manera que su implementación se encuentra distribuida en los clientes, los servidores de aplicaciones y los servidores de bases de datos. En muchos casos los servidores cuentan con mecanismos de tolerancia a fallas y son máquinas multiprocesador con grandes cantidades de memoria y recursos.

El siguiente diagrama ilustra como encaja la CSLA en la arquitectura n-tier

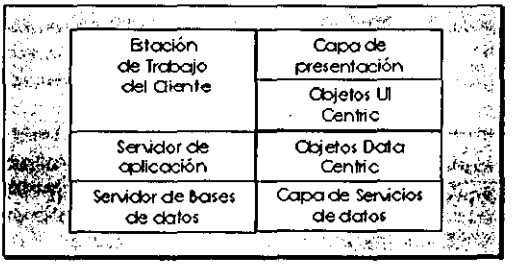


Figura B.3

La UI por si misma junto con los objetos UI-Centric permanecen en la estación de trabajo, esto permite ofrecer al usuario la interfaz mas completa posible, y nos permite tomar ventaja del poder de procesamiento en la estación de trabajo.

Los objetos Data-Centric se encuentran centralizados en un servidor de aplicaciones. Estos objetos se caracterizan por su gran interacción con la base de datos y es muy recomendable que el servidor de aplicaciones cuente con una conexión de gran ancho de banda hacia el servidor de base de datos para incrementar el performance de la aplicación.

En algunos casos resulta necesario que la aplicación interactúe con múltiples fuentes de datos. Por ejemplo, los datos esenciales pueden residir en una base de datos Oracle, pero puede ser necesario obtener información de otro servidor como DB2 o SQL Server. Los objetos Data-Centric que se ejecutan en el servidor de aplicaciones pueden aislar casi por completo a los objetos UI-Centric de la complejidad que resulta del trabajo simultaneo con diversas fuentes de datos.

ARQUITECTURA DE UNA SOLA COMPUTADORA (SINGLE COMPUTER)

El escenario de una sola computadora es importante debido a que no solamente representa a una estación de trabajo ejecutando una aplicación, sino que proporciona la base para aplicaciones desconectadas. Ejemplos de tales aplicaciones con dispositivos desconectados son las que se ejecutan en computadoras portátiles (laptops) y dispositivos aun más pequeños (handled PC´s) Aunque estos dispositivos se encuentran desconectados, no significa que no estarán conectados periódicamente a un servidor para vaciar los datos recolectados u obtener una reprogramación.

Algunos analistas industriales predicen que las aplicaciones desconectadas tendrán más y más importancia en tanto el hardware para handled PC´s, PC´s de automóviles y otros dispositivos sean más comunes.

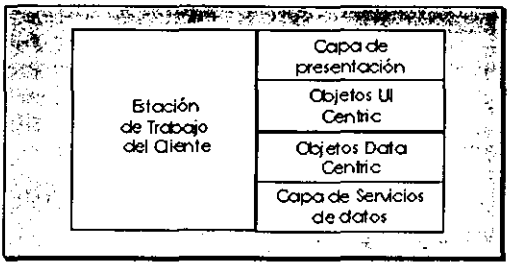


Figura B.4

A pesar de que toda la lógica de la interfaz y de negocios de la aplicación reside en el cliente, esta arquitectura tiene la ventaja de explotar al 100% el poder de procesamiento del cliente, pero tiene la desventaja de tener que instalar la aplicación completa en cada cliente, lo que resulta en un costo de mantenimiento muy alto

ARQUITECTURA BROWSER-BASED

La arquitectura física browser-based ha venido tomando importancia hasta hace apenas pocos años con el surgimiento del World Wide Web (WWW). Las aplicaciones browser-based, en su mayoría han sido diseñadas para tener una interfaz delgada en el cliente, normalmente ejecutándose en un browser. En esta arquitectura, la lógica de negocio y el procesamiento son manipuladas por el servidor Web u otros servidores de aplicaciones invocados por el servidor Web.

En esta arquitectura se tiene la ventaja de eliminar virtualmente cualquier lógica que tenga que ver con la implementación o actualización de las estaciones de trabajo. Sin embargo, esta simplicidad viene a expensas de una interfaz de usuario robusta. Con tales clientes delgados, la única forma de proporcionar una validación, cálculos o procesamiento, es requerirlo al servidor. De esta forma, la estación de trabajo del cliente es básicamente una terminal.

Algunas tecnologías han venido surgiendo para poder dar solución a estos inconvenientes. Estas tecnologías toman ventaja del poder de procesamiento del cliente, manteniendo al mismo tiempo los beneficios de un ambiente basado en

browser. Este modelo de aplicaciones es muy atractivo, dado que permite alcanzar los beneficios del manejo de la lógica de negocios y proporcionar aun al usuario la experiencia de un interfaz robusta.

La arquitectura física de una aplicación inteligente browser-based incluye una estación de trabajo, un servidor web, posiblemente un servidor de aplicaciones y un servidor de base de datos. Es posible mapear la CSLA a este tipo de aplicaciones como se muestra en el siguiente diagrama:

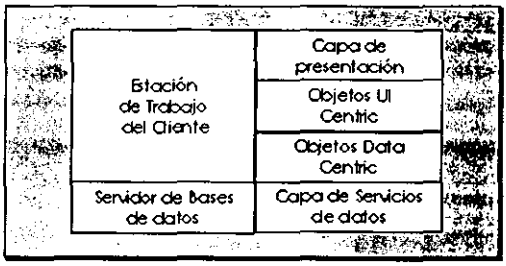


Figura B.5

Esta variante puede ocasionar una carga de trabajo sustancial sobre el servidor de Web, dado que los objetos Data-Centric se encuentran ejecutándose directamente en el servidor, lo que puede limitar la escalabilidad del sistema, de manera que se puede optar por ejecutar los objetos Data-Centric en un servidor de aplicaciones y utilizar el servidor de Web como mero conducto para las comunicaciones entre el cliente y el servidor de aplicaciones.

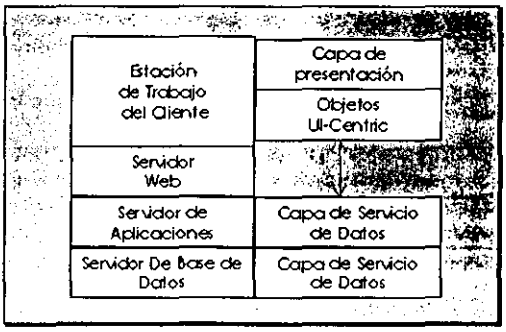


Figura B.6

En cada uno de los casos se depende de las tecnologías del browser para el manejo de la descarga e instalación de los componentes UI-Centric y Data-Centric. Una vez que los componentes se encuentran instalados en la máquina cliente, el browser puede alojar la interfaz de usuario haciendo uso de los objetos UI-Centric para proporcionar al usuario la experiencia de una interfaz robusta.

APÉNDICE C

LISTADOS DEL CASO PRÁCTICO

Listado 1. Creación de la base de datos, tablas y relaciones

```
/* ScriptWindowsDNA_DNS.SQL
**
** AUTOR
**
** FECHA:
**
** OBJETIVO:
**        Este script genera la base de datos, tablas, relaciones,
**        reglas, constraints,triggers,etc. que es la capa de datos del DNS
**        que forma parte del caso práctico de la tesis.
**
** NOTA:
**        Es importante mencionar que este es código que generó ERwin y que fué
**        modificado para que incluyera la creación de la base de datos.
*/

USE master
GO
CREATE DATABASE Tesis
ON
( NAME = Tesis_Dat,
  FILENAME = 'D:\Tesis\Capa de Datos\TesisDat.mdf',
  SIZE = 100,
  MAXSIZE = 200,
  FILEGROWTH = 10)
LOG ON
( NAME = 'Tesis_Log',
  FILENAME = 'D:\Tesis\Capa de Datos\TesisLog.ldf',
  SIZE = 20MB,
  MAXSIZE = 40MB,
  FILEGROWTH = 5MB )
GO

USE Tesis
GO
 CREATE TABLE Compania (
        Cia_Id                  Clave,
        Cia_Desc                DescripcionMedia,
        Cia_Dir                 Direccion,
        Cia_Alias               varchar(20) NOT NULL,
        Cia_Curp                CURP NOT NULL,
        Edo_Id                  Clave,
        Municipio_Id            Clave,
        Cia_Rfc                 RFC,
        Colonia_Id              Clave,
        Cia_Cemp                CURP NOT NULL,
        Cia_Status              Status,
        Cia_Tel                 Telefono,
        Cia_Email               EMail,
        CONSTRAINT XPKCompañia
              PRIMARY KEY (Cia_Id)
 )
go

 CREATE TABLE ContaCtaContable (
        ContaCta_Id             char(20) NOT NULL,
        ContaCta_Desc           varchar(150) NOT NULL,
        ContaCta_Categoria      Tipo,
        ContaCta_Activa         Status,
        ContaCta_FechaAlta      Fecha,
        ContaCta_FechaCambio    Fecha,
        SysUser_Id              int NOT NULL,
        CONSTRAINT XPKContaCtaContable
              PRIMARY KEY (ContaCta_Id)
 )
go

 CREATE TABLE ContaEdoDocto (
        ContaEdoDoc_Id          Clave,
        ContaEdoDoc_Desc        DescripcionCorta,
        ContaEdoDoc_Status      Status,
        CONSTRAINT XPKContaEdoPoliza
              PRIMARY KEY (ContaEdoDoc_Id)
 )
go

 CREATE TABLE ContaHistorialCta (
        ContaCta_Id             char(20) NOT NULL,
        ContaHC_AnoFiscal       Clave,
        ContaHC_Bal01           Moneda,
        ContaHC_Debe01          Moneda,
        ContaHC_Haber01         Moneda,
        ContaHC_Bal02           Moneda,
        ContaHC_Debe02          Moneda,
        ContaHC_Haber02         Moneda,
```

```
        ContaHC_Bal03       Moneda,
        ContaHC_Debe03      Moneda,
        ContaHC_Haber03     Moneda,
        ContaHC_Bal04       Moneda,
        ContaHC_Debe04      Moneda,
        ContaHC_Haber04     Moneda,
        ContaHC_Bal05       Moneda,
        ContaHC_Debe05      Moneda,
        ContaHC_Haber05     Moneda,
        ContaHC_Bal06       Moneda,
        ContaHC_Debe06      Moneda,
        ContaHC_Haber06     Moneda,
        ContaHC_Bal07       Moneda,
        ContaHC_Debe07      Moneda,
        ContaHC_Haber07     Moneda,
        ContaHC_Bal08       Moneda,
        ContaHC_Debe08      Moneda,
        ContaHC_Haber08     Moneda,
        ContaHC_Bal09       Moneda,
        ContaHC_Debe09      Moneda,
        ContaHC_Haber09     Moneda,
        ContaHC_Bal10       Moneda,
        ContaHC_Debe10      Moneda,
        ContaHC_Haber10     Moneda,
        ContaHC_Bal11       Moneda,
        ContaHC_Debe11      Moneda,
        ContaHC_Haber11     Moneda,
        ContaHC_Bal12       Moneda,
        ContaHC_Debe12      Moneda,
        ContaHC_Haber12     Moneda,
        ContaHC_Bal13       Moneda,
        CONSTRAINT XPKContaHistorialCta
              PRIMARY KEY (ContaCta_Id, ContaHC_AnoFiscal)
)
go

CREATE TABLE ContaHistorialCtaHist (
        ContaCta_Id         char(20) NOT NULL,
        ContaHC_AnoFiscal   Clave,
        ContaHC_Bal01       Moneda,
        ContaHC_Debe01      Moneda,
        ContaHC_Haber01     Moneda,
        ContaHC_Bal02       Moneda,
        ContaHC_Debe02      Moneda,
        ContaHC_Haber02     Moneda,
        ContaHC_Bal03       Moneda,
        ContaHC_Debe03      Moneda,
        ContaHC_Haber03     Moneda,
        ContaHC_Bal04       Moneda,
        ContaHC_Debe04      Moneda,
        ContaHC_Haber04     Moneda,
        ContaHC_Bal05       Moneda,
        ContaHC_Debe05      Moneda,
        ContaHC_Haber05     Moneda,
        ContaHC_Bal06       Moneda,
        ContaHC_Debe06      Moneda,
        ContaHC_Haber06     Moneda,
        ContaHC_Bal07       Moneda,
        ContaHC_Debe07      Moneda,
        ContaHC_Haber07     Moneda,
        ContaHC_Bal08       Moneda,
        ContaHC_Debe08      Moneda,
        ContaHC_Haber08     Moneda,
        ContaHC_Bal09       Moneda,
        ContaHC_Debe09      Moneda,
        ContaHC_Haber09     Moneda,
        ContaHC_Bal10       Moneda,
        ContaHC_Debe10      Moneda,
        ContaHC_Haber10     Moneda,
        ContaHC_Bal11       Moneda,
        ContaHC_Debe11      Moneda,
        ContaHC_Haber11     Moneda,
        ContaHC_Bal12       Moneda,
        ContaHC_Debe12      Moneda,
        ContaHC_Haber12     Moneda,
        ContaHC_Bal13       Moneda,
        CONSTRAINT XPKContaHistorialCta
              PRIMARY KEY (ContaCta_Id, ContaHC_AnoFiscal)
)
go

CREATE TABLE ContaPeriodo (
        Cia_Id              Clave,
        Periodo_Id          char(6) NOT NULL,
        ContaPer_Contabilidadi char(6) NOT NULL,
        ContaPer_CxP        char(6) NOT NULL,
        ContaPer_Status     Status,
        CONSTRAINT XPKContaPeriodo
              PRIMARY KEY (Cia_Id, Periodo_Id)
)
go

CREATE TABLE ContaPoliza (
        Cia_Id              Clave,
        Reg_Id              Clave,
```

```
        Fra_Id                  Clave,
        TipoPoliza_Id           Clave,
        Periodo_Id              char(6) NOT NULL,
        ContaPol_Id             int NOT NULL,
        ContaPol_Concepto       DescripcionCorta,
        ContaPol_FechaCon       Fecha,
        ContaPol_FechaAlta      Fecha,
        ContaPol_FechaCambio    Fecha,
        ContaPol_TotalPol       Moneda,
        ContaEdoDoc_Id          Clave,
        SysModulo_Id            int NOT NULL,
        SysUser_Id              int NOT NULL,
        CONSTRAINT XPKContaPoliza
              PRIMARY KEY (Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id,
              Periodo_Id, ContaPol_Id)
 )
 go

 CREATE TABLE ContaPolizaDet (
        Cia_Id                  Clave,
        Reg_Id                  Clave,
        Fra_Id                  Clave,
        TipoPoliza_Id           Clave,
        Periodo_Id              char(6) NOT NULL,
        ContaPol_Id             int NOT NULL,
        ContaPol_NoMov          int NOT NULL,
        ContaCta_Id             char(20) NOT NULL,
        ContaPol_Concepto       DescripcionCorta,
        ContaPol_Debe           Moneda,
        ContaPol_Haber          Moneda,
        CONSTRAINT XPKContaPolizaDet
              PRIMARY KEY (Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id,
              Periodo_Id, ContaPol_Id, ContaPol_NoMov)
 )
go

 CREATE TABLE ContaPolizaDetHist (
        Cia_Id                  Clave,
        Reg_Id                  Clave,
        Fra_Id                  Clave,
        TipoPoliza_Id           Clave,
        Periodo_Id              char(6) NOT NULL,
        ContaPol_Id             int NOT NULL,
        ContaPol_NoMov          int NOT NULL,
        ContaCta_Id             char(20) NOT NULL,
        ContaPol_Concepto       DescripcionCorta,
        ContaPol_Debe           Moneda,
        ContaPol_Haber          Moneda,
        CONSTRAINT XPKContaPolizaDet
              PRIMARY KEY (Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id,
              Periodo_Id, ContaPol_Id, ContaPol_NoMov)
 )
go

 CREATE TABLE ContaPolizaHist (
        Cia_Id                  Clave,
        Reg_Id                  Clave,
        Fra_Id                  Clave,
        TipoPoliza_Id           Clave,
        Periodo_Id              char(6) NOT NULL,
        ContaPol_Id             int NOT NULL,
        ContaPol_Concepto       DescripcionCorta,
        ContaPol_FechaCon       Fecha,
        ContaPol_FechaAlta      Fecha,
        ContaPol_FechaCambio    Fecha,
        ContaPol_TotalPol       Moneda,
        ContaEdoDoc_Id          Clave,
        SysModulo_Id            int NOT NULL,
        SysUser_Id              int NOT NULL,
        CONSTRAINT XPKContaPoliza
              PRIMARY KEY (Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id,
              Periodo_Id, ContaPol_Id)
 )
go

 CREATE TABLE Fraccionamiento (
        Fra_Id                  Clave,
        Fra_Desc                DescripcionMedia,
        Fra_Dir                 Direccion,
        Edo_Id                  Clave,
        Municipio_Id            Clave,
        Fra_Tel                 Telefono NOT NULL,
        Colonia_Id              Clave,
        Fra_Email               EMail,
        Fra_Status              Status,
        Fra_Ini                 Clave,
        Reg_Id                  Clave,
        ZonaCob_Id              Clave,
        Iva_Id                  Clave,
        CONSTRAINT XPKFraccionamiento
              PRIMARY KEY (Fra_Id)
 )
go

 CREATE TABLE Region (
```

```
        Reg_Id              Clave,
        Reg_Desc            DescripcionCorta,
        Reg_Status          Status,
        CONSTRAINT XPKRegion
              PRIMARY KEY (Reg_Id)
 )
go

 CREATE TABLE SysModulo (
        SysModulo_Id        int NOT NULL,
        SysModulo_Desc      DescripcionCorta,
        SysModulo_Status    Status,
        SysModulo_Orden     int NOT NULL,
        CONSTRAINT XPKModulos
              PRIMARY KEY (SysModulo_Id)
 )
go

 CREATE TABLE SysUsuario (
        SysUser_Id          int NOT NULL,
        SysUser_Usuario     varchar(10) NOT NULL,
        SysUser_Nombre      DescripcionCorta,
        SysUser_ApePat      Apellido,
        SysUser_ApeMat      Apellido,
        SysUser_OldPwd      Password,
        SysUser_NewPwd      Password,
        SysUser_Status      Status,
        CONSTRAINT XPKUsuarios
              PRIMARY KEY (SysUser_Id)
 )
go

 CREATE TABLE TipoPoliza (
        TipoPoliza_Id       Clave,
        TipoPoliza_Desc     DescripcionCorta,
        TipoPoliza_Status   Status,
        CONSTRAINT XPKTipoPoliza
              PRIMARY KEY (TipoPoliza_Id)
 )
go

 ALTER TABLE Compania
        ADD CONSTRAINT R_822
              FOREIGN KEY (Edo_Id, Municipio_Id, Colonia_Id)
                    REFERENCES Colonia
go

 ALTER TABLE ContaCtaContable
        ADD CONSTRAINT R_1164
              FOREIGN KEY (SysUser_Id)
                    REFERENCES SysUsuario
go

 ALTER TABLE ContaHistorialCta
        ADD CONSTRAINT R_1208
              FOREIGN KEY (ContaCta_Id)
                    REFERENCES ContaCtaContable
go

 ALTER TABLE ContaHistorialCtaHist
        ADD CONSTRAINT R_1209
              FOREIGN KEY (ContaCta_Id)
                    REFERENCES ContaCtaContable
go

 ALTER TABLE ContaPeriodo
        ADD CONSTRAINT R_1154
              FOREIGN KEY (Cia_Id)
                    REFERENCES Compania
go

 ALTER TABLE ContaPoliza
        ADD CONSTRAINT R_1162
              FOREIGN KEY (SysUser_Id)
                    REFERENCES SysUsuario
go

 ALTER TABLE ContaPoliza
        ADD CONSTRAINT R_1161
              FOREIGN KEY (SysModulo_Id)
                    REFERENCES SysModulo
go

 ALTER TABLE ContaPoliza
        ADD CONSTRAINT R_1160
              FOREIGN KEY (ContaRdoDoc_Id)
                    REFERENCES ContaEdoDocto
go

 ALTER TABLE ContaPoliza
        ADD CONSTRAINT R_1158
              FOREIGN KEY (TipoPoliza_Id)
                    REFERENCES TipoPoliza
go
```

```
 ALTER TABLE ContaPoliza
       ADD CONSTRAINT R_1156
             FOREIGN KEY (Reg_Id)
                   REFERENCES Region
go

 ALTER TABLE ContaPolizaDet
       ADD CONSTRAINT R_1165
             FOREIGN KEY (ContaCta_Id)
                   REFERENCES ContaCtaContable
go

 ALTER TABLE ContaPolizaDet
       ADD CONSTRAINT R_1163
             FOREIGN KEY (Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id,
             Periodo_Id, ContaPol_Id)
                   REFERENCES ContaPoliza
go

 ALTER TABLE ContaPolizaDetHist
       ADD CONSTRAINT R_1222
             FOREIGN KEY (ContaCta_Id)
                   REFERENCES ContaCtaContable
go

 ALTER TABLE ContaPolizaDetHist
       ADD CONSTRAINT R_1221
             FOREIGN KEY (Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id,
             Periodo_Id, ContaPol_Id)
                   REFERENCES ContaPolizaHist
go

 ALTER TABLE ContaPolizaHist
       ADD CONSTRAINT R_1249
             FOREIGN KEY (ContaEdoDoc_Id)
                   REFERENCES ContaEdoDocto
go

 ALTER TABLE ContaPolizaHist
       ADD CONSTRAINT R_1220
             FOREIGN KEY (TipoPoliza_Id)
                   REFERENCES TipoPoliza
go

 ALTER TABLE ContaPolizaHist
       ADD CONSTRAINT R_1219
             FOREIGN KEY (Reg_Id)
                   REFERENCES Region
go

 ALTER TABLE ContaPolizaHist
       ADD CONSTRAINT R_1217
             FOREIGN KEY (SysUser_Id)
                   REFERENCES SysUsuario
go

 ALTER TABLE ContaPolizaHist
       ADD CONSTRAINT R_1216
             FOREIGN KEY (SysModulo_Id)
                   REFERENCES SysModulo
go

 ALTER TABLE Fraccionamiento
       ADD CONSTRAINT R_1143
             FOREIGN KEY (Reg_Id)
                   REFERENCES Region
go
```

Listado 2. Objetos Ejecutantes

Dat_Contabilidad.Insert

```
Option Explicit

Public Sub ContaCtaContable(ByVal ContaCta_Id As String, ByVal ContaCta_Desc As String, _
    ByVal ContaCta_Categoria As String, ByVal ContaCta_Activa As String, _
    ByVal ContaCta_FechaAlta As String, ByVal ContaCta_FechaCambio As String, _
    ByVal SysUser_Id As String)
' Autor:
' Fecha:
' Parametros:
'       ContaCta_Id             -   Id de la Cuenta
'       ContaCta_Desc           -   Descripcion de la cuenta
'       ContaCta_Categoria      -   Categoria de la cuenta (A,D)
'       ContaCta_Activa         -   Estado de la Cuenta
'       ContaCta_FechaAlta      -   Fecha de alta
'       ContaCta_FechaCambio    -   Fecha de cambio
'       SysUser_Id              -   Id del usuario
'
' Objetivo:
'       Insertar fisicamente el registro en la base de datos

Dim ContaCta_Origen As String
```

```
Dim adoCon As ADODB.Connection
Dim adoErr As ADODB.Error
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext

    Set adoCon = CreateObject("ADODB.Connection")
    'Prepara la sentencia
        str_SQL = "Insert Into ContaCtaContable

(ContaCta_Id,ContaCta_Desc,ContaCta_Categoria," & _
            "ContaCta_Activa,ContaCta_FechaAlta,ContaCta_FechaCambio,SysUser_Id) Values ('"

& _
            ContaCta_Id & "','" & ContaCta_Desc & "','" & ContaCta_Categoria & "'," & _
            ContaCta_Activa & ",'" & ContaCta_FechaAlta & "','" & ContaCta_FechaCambio &

"'," & SysUser_Id & ")"
    adoCon.ConnectionString = adoConString
    adoCon.Open
    adoCon.Execute str_SQL
    adoCon.Close
    Set adoCon = Nothing

    'Acepta la transacción Banco
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    Set objContext = Nothing
    If adoCon Is Nothing Then   'Esta cerrada y es de VB
        Err.Raise Err.Number & " ", "dat_Contabilidad.Insert.ContaCtaContable",

Err.Description
    Else        'Esta abierta
        If adoCon.Errors.Count = 0 Then     'Esta abierta y es de VB
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise Err.Number & " ", "dat_Contabilidad.Insert.ContaCtaContable",

Err.Description
            Exit Sub
        Else        'Esta abierta y es de DB
            For Each adoErr In adoCon.Errors
                If adoErr.NativeError = 2627 Then
                    str_ErrDesc = str_ErrDesc & " " & "Violación de la llave primaria"
                Else
                    str_ErrDesc = str_ErrDesc & " " & adoErr.Description
                End If
            Next
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise 10001, "dat_Contabilidad.Insert.ContaCtaContable", str_ErrDesc
        End If
    End If
End Sub


Public Sub ContaPoliza(ByVal Cia_Id As String, ByVal Reg_Id As String, ByVal Fra_Id As

String, ByVal TipoPoliza_Id As String, ByVal Periodo_Id As String, ByVal ContaPol_Id As

String, _
    ByVal ContaPol_Concepto As String, ByVal ContaPol_FechaCon As String, ByVal

ContaPol_FechaAlta As String, ByVal ContaPol_FechaCambio As String, ByVal ContaPol_TotalPol

As String, _
    ByVal ContaEdoDoc_Id As String, ByVal SysModulo_Id As String, ByVal SysUser_Id As

String)
' Autor:
' Fecha:
' Parametros:
'       Cia_Id                  =   Id de la Comañia
'       Reg_Id                  =   Id de la Foranea
'       Fra_Id                  =   Id del Fraccionamiento
'       TipoPoliza_Id           =   Tipo de Poliza (IG,EG,DR,CP,TE,etc.)
'       Periodo_Id              =   Periodo contable
'       ContaPol_Id             =   Numero de Poliza contable
'       ContaPol_Concepto       =   Concepto de la Poliza
'       ContaPol_FechaCon       =   Fecha de contabilización
'       ContaPol_FechaAlta      =   Fecha de captura en el sistema
'       ContaPol_FechaCambio    =   Fecha en que ha sido modificada
'       ContaPol_TotalPol       =   Total de cargos y abonos
'       ContaEdoDoc_Id          =   Estado del documento (BALANCEADA,CANCELADA,etc.)
'       SysModulo_Id            =   Id del modulo que genero esta poliza
'       SysUser_Id              =   Id del usuario
'
' Objetivo:
'       Insertar físicamente el registro en la base de datos
```

```
Dim adoCon As ADODB.Connection
Dim adoErr As ADODB.Error
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext

    Set adoCon = CreateObject("ADODB.Connection")
    'Prepara la sentencia
    str_SQL = "Insert Into ContaPoliza (" & _
    "Cia_Id,Reg_Id,Fra_Id,TipoPoliza_Id,Periodo_Id,ContaPol_Id,ContaPol_Concepto," & _
    "ContaPol_FechaCon,ContaPol_FechaAlta,ContaPol_FechaCambio,ContaPol_TotalPol," & _
    "ContaEdoDoc_Id,SysModulo_Id,SysUser_Id) Values ('" & _
    Cia_Id & "','" & Reg_Id & "','" & Fra_Id & "','" & TipoPoliza_Id & "','" & _
    Periodo_Id & "'," & ContaPol_Id & ",'" & ContaPol_Concepto & "','" & ContaPol_FechaCon &

"','" & _
    ContaPol_FechaAlta & "','" & ContaPol_FechaCambio & "'," & ContaPol_TotalPol & ",'" & _
    ContaEdoDoc_Id & "'," & SysModulo_Id & "," & SysUser_Id & ")"

    adoCon.ConnectionString = adoConString
    adoCon.Open
    adoCon.Execute str_SQL
    adoCon.Close
    Set adoCon = Nothing

    'Acepta la transacción Banco
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    Set objContext = Nothing
    If adoCon Is Nothing Then   'Esta cerrada y es de VB
        Err.Raise Err.Number & " ", "dat_Contabilidad.Insert.ContaPoliza", Err.Description
    Else        'Esta abierta
        If adoCon.Errors.Count = 0 Then     'Esta abierta y es de VB
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise Err.Number & " ", "dat_Contabilidad.Insert.ContaPoliza",

Err.Description
            Exit Sub
        Else        'Esta abierta y es de DB
            For Each adoErr In adoCon.Errors
                If adoErr.NativeError = 2627 Then
                    str_ErrDesc = str_ErrDesc & " " & "Violación de la llave primaria"
                Else
                    str_ErrDesc = str_ErrDesc & " " & adoErr.Description
                End If
            Next
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise 10001, "dat_Contabilidad.Insert.ContaPoliza", str_ErrDesc
        End If
    End If
End Sub

Public Sub ContaPolizaDet(ByVal Cia_Id As String, ByVal Reg_Id As String, ByVal Fra_Id As

String, ByVal TipoPoliza_Id As String, ByVal Periodo_Id As String, ByVal ContaPol_Id As

String, ByVal ContaPol_NoMov As String, _
    ByVal ContaCta_Id As String, ByVal ContaPol_Concepto As String, ByVal ContaPol_Debe As

String, ByVal ContaPol_Haber As String)

' Autor:
' Fecha:
' Parametros:
'       Cia_Id                  =   Id de la Comañia
'       Reg_Id                  =   Id de la Foranea
'       Fra_Id                  =   Id del Fraccionamiento
'       TipoPoliza_Id           =   Tipo de Poliza (IG,EG,DR,CP,TE,etc.)
'       Periodo_Id              =   Periodo contable
'       ContaPol_Id             =   Numero de Poliza contable
'       ContaPol_NoMov          =   Numero de Movimiento
'       ContaCta_Id             =   CuentaContable
'       ContaPol_Concepto       =   Concepto del movimiento
'       ContaPol_Debe           =   Debe
'       ContaPol_Haber          =   Haber
'
' Objetivo:
'       Insertar fisicamente el registro en la base de datos

Dim adoCon As ADODB.Connection
Dim adoErr As ADODB.Error
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext

    Set adoCon = CreateObject("ADODB.Connection")
    'Prepara la sentencia
    str_SQL = "Insert Into ContaPolizaDet
```

```
(Cia_Id,Reg_Id,Fra_Id,TipoPoliza_Id,Periodo_Id,ContaPol_Id," & _
        "ContaPol_NoMov,ContaCta_Id,ContaPol_Concepto,ContaPol_Debe,ContaPol_Haber) Values

(" & _
        Cia_Id & "','" & Reg_Id & "','" & Fra_Id & "','" & TipoPoliza_Id & "','" & _
        Periodo_Id & "'," & ContaPol_Id & "," & ContaPol_NoMov & ",'" & ContaCta_Id & "','"

& _
        ContaPol_Concepto & "'," & ContaPol_Debe & "," & ContaPol_Haber & ")"

    adoCon.ConnectionString = adoConString
    adoCon.Open
    adoCon.Execute str_SQL
    adoCon.Close
    Set adoCon = Nothing

    'Acepta la transacción Banco
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    Set objContext = Nothing
    If adoCon Is Nothing Then   'Esta cerrada y es de VB
        Err.Raise Err.Number & " ", "dat_Contabilidad.Insert.ContaPolizaDet",

Err.Description
    Else        'Esta abierta
        If adoCon.Errors.Count = 0 Then     'Esta abierta y es de VB
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise Err.Number & " ", "dat_Contabilidad.Insert.ContaPolizaDet",

Err.Description
            Exit Sub
        Else        'Esta abierta y es de DB
            For Each adoErr In adoCon.Errors
                If adoErr.NativeError = 2627 Then
                    str_ErrDesc = str_ErrDesc & " " & "Violación de la llave primaria"
                Else
                    str_ErrDesc = str_ErrDesc & " " & adoErr.Description
                End If
            Next
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise 10001, "dat_Contabilidad.Insert.ContaPolizaDet", str_ErrDesc
        End If
    End If
End Sub

Dat_Contabilidad.Update

Public Sub ContaCtaContable(ByVal ContaCta_Id As String, _
    Optional ByVal ContaCta_Desc As String, Optional ByVal ContaCta_Categoria As String, _
    Optional ByVal ContaCta_Activa As String, Optional ByVal ContaCta_FechaCambio As String,

_
    Optional ByVal SysUser_Id As String)
' Autor:
' Fecha:
' Parametros:
'       ContaCta_Id             =   Id de la Cuenta
'       ContaCta_Desc           =   Descripcion de la cuenta
'       ContaCta_Categoria      =   Categoria de la cuenta
'       ContaCta_Activa         =   Estado de la Cuenta
'       ContaCta_FechaCambio    =   Fecha de cambio
'       SysUser_Id              =   Id del usuario
'
' Objetivo:
'       Insertar físicamente el registro en la base de datos

Dim adoCon As ADODB.Connection
Dim adoErr As ADODB.Error
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext
    'Prepara la sentencia

        str_SQL = "Update ContaCtaContable Set "
        If ContaCta_Desc <> "" Then str_SQL = str_SQL & "ContaCta_Desc='" & ContaCta_Desc &

"',"
        If ContaCta_Activa <> "" Then str_SQL = str_SQL & "ContaCta_Activa=" &

ContaCta_Activa & ","
        If ContaCta_Categoria <> "" Then str_SQL = str_SQL & "ContaCta_Categoria='" &

ContaCta_Categoria & "',"
        If ContaCta_FechaCambio <> "" Then str_SQL = str_SQL & "ContaCta_FechaCambio='" &

ContaCta_FechaCambio & "',"
        If SysUser_Id <> "" Then str_SQL = str_SQL & "SysUser_Id=" & SysUser_Id & ","
        'Quita la coma de al final
        str_SQL = Left$(str_SQL, Len(str_SQL) - 1)
```

```
        str_SQL = str_SQL & " Where ContaCta_Id='" & ContaCta_Id & "'"

    Set adoCon = CreateObject("ADODB.Connection")
    adoCon.ConnectionString = adoConString
    adoCon.Open
    adoCon.Execute str_SQL
    adoCon.Close
    Set adoCon = Nothing

    'Acepta la transacción Banco
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    Set objContext = Nothing
    If adoCon Is Nothing Then   'Esta cerrada y es de VB
        Err.Raise Err.Number & " ", "dat_Contabilidad.Update.ContaCtaContable",

Err.Description
    Else        'Esta abierta
        If adoCon.Errors.Count = 0 Then     'Esta abierta y es de VB
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise Err.Number & " ", "dat_Contabilidad.Update.ContaCtaContable",

Err.Description
            Exit Sub
        Else        'Esta abierta y es de DB
            For Each adoErr In adoCon.Errors
                str_ErrDesc = str_ErrDesc & " " & adoErr.Description
            Next
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise 10001, "dat_Contabilidad.Update.ContaCtaContable", str_ErrDesc
        End If
    End If
End Sub

Public Sub ContaHistorialCtaCA(ByVal ContaCta_Id As String, ByVal Periodo_Id, ByVal

ContaHC_Debe As String, ByVal ContaHC_Haber As String)
' Autor:
' Fecha:
' Parametros:
'       ContaCta_Id         =   Id de la ContaCta_Id
'       Periodo_Id          =   Periodo Contable
'       ContaHC_Debe        =   ContaHC_Debe
'       ContaHC_Haber       =   ContaHC_Haber
'
' Objetivo:
'       Actualiza la tabla de ContaHistorialCta en todos los campos de ContaHC_Baln
'       la actualización la hace respecto al cargo y el abono sobre la cuenta.
'       Esta funcion se ejecuta cada vez que se carga o se abona sobre una cuenta

Dim cur_Diferencia As Currency
Dim adoCon As ADODB.Connection
Dim adoErr As ADODB.Error
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext
    'Obtiene la diferencia del cargo y el abono
    cur_Diferencia = ContaHC_Debe - ContaHC_Haber
    'Prepara la sentencia
    Select Case Right$(Periodo_Id, 2)
    Case "01"
    str_SQL = "Update ContaHistorialCta Set " & _
            "ContaHC_Debe01 = ContaHC_Debe01 + (" & ContaHC_Debe & "),* & _
            "ContaHC_Haber01 = ContaHC_Haber01 + (" & ContaHC_Haber & "),* & _
            "ContaHC_Bal02 = ContaHC_Bal02 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal03 = ContaHC_Bal03 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal04 = ContaHC_Bal04 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal05 = ContaHC_Bal05 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal06 = ContaHC_Bal06 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal07 = ContaHC_Bal07 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal08 = ContaHC_Bal08 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal09 = ContaHC_Bal09 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal10 = ContaHC_Bal10 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

    Case "02"
    str_SQL = "Update ContaHistorialCta Set " & _
            "ContaHC_Debe02 = ContaHC_Debe02 + (" & ContaHC_Debe & "),* & _
            "ContaHC_Haber02 = ContaHC_Haber02 + (" & ContaHC_Haber & "),* & _
            "ContaHC_Bal03 = ContaHC_Bal03 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal04 = ContaHC_Bal04 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal05 = ContaHC_Bal05 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal06 = ContaHC_Bal06 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal07 = ContaHC_Bal07 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal08 = ContaHC_Bal08 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal09 = ContaHC_Bal09 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal10 = ContaHC_Bal10 + (" & CStr(cur_Diferencia) & "),* & _
```

```
        "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

Case "03"
str_SQL = "Update ContaHistorialCta Set " & _
        "ContaHC_Debe03 = ContaHC_Debe03 + (" & ContaHC_Debe & "),* & _
        "ContaHC_Haber03 = ContaHC_Haber03 + (" & ContaHC_Haber & "),* & _
        "ContaHC_Bal04 = ContaHC_Bal04 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal05 = ContaHC_Bal05 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal06 = ContaHC_Bal06 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal07 = ContaHC_Bal07 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal08 = ContaHC_Bal08 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal09 = ContaHC_Bal09 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal10 = ContaHC_Bal10 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

Case "04"
str_SQL = "Update ContaHistorialCta Set " & _
        "ContaHC_Debe04 = ContaHC_Debe04 + (" & ContaHC_Debe & "),* & _
        "ContaHC_Haber04 = ContaHC_Haber04 + (" & ContaHC_Haber & "),* & _
        "ContaHC_Bal05 = ContaHC_Bal05 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal06 = ContaHC_Bal06 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal07 = ContaHC_Bal07 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal08 = ContaHC_Bal08 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal09 = ContaHC_Bal09 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal10 = ContaHC_Bal10 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

Case "05"
str_SQL = "Update ContaHistorialCta Set " & _
        "ContaHC_Debe05 = ContaHC_Debe05 + (" & ContaHC_Debe & "),* & _
        "ContaHC_Haber05 = ContaHC_Haber05 + (" & ContaHC_Haber & "),* & _
        "ContaHC_Bal06 = ContaHC_Bal06 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal07 = ContaHC_Bal07 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal08 = ContaHC_Bal08 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal09 = ContaHC_Bal09 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal10 = ContaHC_Bal10 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

Case "06"
str_SQL = "Update ContaHistorialCta Set " & _
        "ContaHC_Debe06 = ContaHC_Debe06 + (" & ContaHC_Debe & "),* & _
        "ContaHC_Haber06 = ContaHC_Haber06 + (" & ContaHC_Haber & "),* & _
        "ContaHC_Bal07 = ContaHC_Bal07 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal08 = ContaHC_Bal08 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal09 = ContaHC_Bal09 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal10 = ContaHC_Bal10 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

Case "07"
str_SQL = "Update ContaHistorialCta Set " & _
        "ContaHC_Debe07 = ContaHC_Debe07 + (" & ContaHC_Debe & "),* & _
        "ContaHC_Haber07 = ContaHC_Haber07 + (" & ContaHC_Haber & "),* & _
        "ContaHC_Bal08 = ContaHC_Bal08 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal09 = ContaHC_Bal09 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal10 = ContaHC_Bal10 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

Case "08"
str_SQL = "Update ContaHistorialCta Set " & _
        "ContaHC_Debe08 = ContaHC_Debe08 + (" & ContaHC_Debe & "),* & _
        "ContaHC_Haber08 = ContaHC_Haber08 + (" & ContaHC_Haber & "),* & _
        "ContaHC_Bal09 = ContaHC_Bal09 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal10 = ContaHC_Bal10 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

Case "09"
str_SQL = "Update ContaHistorialCta Set " & _
        "ContaHC_Debe09 = ContaHC_Debe09 + (" & ContaHC_Debe & "),* & _
        "ContaHC_Haber09 = ContaHC_Haber09 + (" & ContaHC_Haber & "),* & _
        "ContaHC_Bal10 = ContaHC_Bal10 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

Case "10"
str_SQL = "Update ContaHistorialCta Set " & _
        "ContaHC_Debe10 = ContaHC_Debe10 + (" & ContaHC_Debe & "),* & _
        "ContaHC_Haber10 = ContaHC_Haber10 + (" & ContaHC_Haber & "),* & _
        "ContaHC_Bal11 = ContaHC_Bal11 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
        "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "
```

```
    Case "11"
    str_SQL = "Update ContaHistorialCta Set " & _
            "ContaHC_Debe11 = ContaHC_Debe11 + (" & ContaHC_Debe & "),* & _
            "ContaHC_Haber11 = ContaHC_Haber11 + (" & ContaHC_Haber & "),* & _
            "ContaHC_Bal12 = ContaHC_Bal12 + (" & CStr(cur_Diferencia) & "),* & _
            "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "

    Case "12"
    str_SQL = "Update ContaHistorialCta Set " & _
            "ContaHC_Debe12 = ContaHC_Debe12 + (" & ContaHC_Debe & "),* & _
            "ContaHC_Haber12 = ContaHC_Haber12 + (" & ContaHC_Haber & "),* & _
            "ContaHC_Bal13 = ContaHC_Bal13 + (" & CStr(cur_Diferencia) & ") "
    End Select

    'Verifica el nivel de la cuenta  y compone la sentencia para que la aplicacion
    'del cargo y el abono incluya a las cuentas acumulativas
    'Verifica si es segundo nivel
    If Right$(ContaCta_Id, 11) = "00000000000" Then
        str_SQL = str_SQL & _
        "Where ContaCta_Id= '" & Left$(ContaCta_Id, 5) & "000000000000000' or " & _
        "ContaCta_Id='" & ContaCta_Id & "'"

    'Verifica si es tercer nivel
    ElseIf Right$(ContaCta_Id, 7) = "0000000" Then
        str_SQL = str_SQL & _
        "Where ContaCta_Id= '" & Left$(ContaCta_Id, 5) & "000000000000000' or " & _
        "ContaCta_Id= '" & Left$(ContaCta_Id, 9) & "00000000000' or " & _
        "ContaCta_Id= '" & ContaCta_Id & "'"
    'Verifica si es cuarto nivel
    ElseIf Right$(ContaCta_Id, 3) = "000" Then
        str_SQL = str_SQL & _
        "Where ContaCta_Id= '" & Left$(ContaCta_Id, 5) & "000000000000000' or " & _
        "ContaCta_Id= '" & Left$(ContaCta_Id, 9) & "00000000000' or " & _
        "ContaCta_Id= '" & Left$(ContaCta_Id, 13) & "0000000' or " & _
        "ContaCta_Id= '" & ContaCta_Id & "'"
    'Verifica si es quinto nivel
    Else
        str_SQL = str_SQL & _
        "Where ContaCta_Id= '" & Left$(ContaCta_Id, 5) & "000000000000000' or " & _
        "ContaCta_Id= '" & Left$(ContaCta_Id, 9) & "00000000000' or " & _
        "ContaCta_Id= '" & Left$(ContaCta_Id, 13) & "0000000' or " & _
        "ContaCta_Id= '" & Left$(ContaCta_Id, 17) & "000' or " & _
        "ContaCta_Id= '" & ContaCta_Id & "'"
    End If

    Set adoCon = CreateObject("ADODB.Connection")
    adoCon.ConnectionString = adoConString
    adoCon.Open
    adoCon.Execute str_SQL
    adoCon.Close
    Set adoCon = Nothing

    'Acepta la transacción Banco
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    Set objContext = Nothing
    If adoCon Is Nothing Then   'Esta cerrada y es de VB
        Err.Raise Err.Number, "dat_Contabilidad.Update.ContaHistorialCtaCA", Err.Description
    Else        'Esta abierta
        If adoCon.Errors.Count = 0 Then     'Esta abierta y es de VB
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise Err.Number, "dat_Contabilidad.Update.ContaHistorialCtaCA",

Err.Description
            Exit Sub
        Else        'Esta abierta y es de DB
            For Each adoErr In adoCon.Errors
                str_ErrDesc = str_ErrDesc & " " & adoErr.Description
            Next
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise 10001, "dat_Contabilidad.Update.ContaHistorialCtaCA", str_ErrDesc
        End If
    End If
End Sub

Public Sub ContaPoliza(ByVal SysUser_Id As String, ByVal Cia_Id As String, ByVal Reg_Id As

String, ByVal Fra_Id As String, _
    ByVal TipoPoliza_Id As String, ByVal Periodo_Id As String, ByVal ContaPol_Id As String,


_
    Optional ByVal ContaPol_Concepto As String, Optional ByVal ContaPol_FechaCon As String,

Optional ByVal ContaPol_FechaCambio As String, _
    Optional ByVal ContaPol_TotalPol As String, Optional ByVal ContaEdoDoc_Id As String)
' Autor:
' Fecha:
' Parametros:
```

```
'       Cia_Id                  =   Id de la Comañia
'       Reg_Id                  =   Id de la Foranea
'       Fra_Id                  =   Id del Fraccionamiento
'       TipoPoliza_Id           =   Tipo de Poliza (IG,EG,DR,CP,TE,etc.)
'       Periodo_Id              =   Periodo contable
'       ContaPol_Id             =   Numero de Poliza contable
'       ContaPol_Concepto       =   Concepto de la Poliza
'       ContaPol_FechaCon       =   Fecha de contabilización
'       ContaPol_FechaCambio    =   Fecha en que ha sido modificada
'       ContaPol_TotalPol       =   Total de cargos y abonos
'       ContaEdoDoc_Id          =   Estado del documento (BALANCEADA,CANCELADA,etc.)
'       SysUser_Id              =   Id del usuario
'
' Objetivo:
'       Actualiza físicamente el registro en la base de datos

Dim adoCon As ADODB.Connection
Dim adoErr As ADODB.Error
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext

    'Prepara la sentencia
    str_SQL = "Update ContaPoliza Set "
    If ContaPol_Concepto <> "" Then str_SQL = str_SQL & "ContaPol_Concepto = '" &

ContaPol_Concepto & "',"
    If ContaPol_FechaCon <> "" Then str_SQL = str_SQL & "ContaPol_FechaCon = '" &

ContaPol_FechaCon & "',"
    If ContaPol_FechaCambio <> "" Then str_SQL = str_SQL & "ContaPol_FechaCambio = '" &

ContaPol_FechaCambio & "',"
    If ContaPol_TotalPol <> "" Then str_SQL = str_SQL & "ContaPol_TotalPol = " &

ContaPol_TotalPol & ","
    If ContaEdoDoc_Id <> "" Then str_SQL = str_SQL & "ContaEdoDoc_Id ='" & ContaEdoDoc_Id &

"',"
    str_SQL = Left(str_SQL, Len(str_SQL) - 1) & " Where " & _
    "Cia_Id='" & Cia_Id & "' and " & _
    "Reg_Id='" & Reg_Id & "' and " & _
    "Fra_Id='" & Fra_Id & "' and " & _
    "TipoPoliza_Id='" & TipoPoliza_Id & "' and " & _
    "Periodo_Id='" & Periodo_Id & "' and " & _
    "ContaPol_Id=" & ContaPol_Id

    Set adoCon = CreateObject("ADODB.Connection")
    adoCon.ConnectionString = adoConString
    adoCon.Open
    adoCon.Execute str_SQL
    adoCon.Close
    Set adoCon = Nothing

    'Acepta la transacción Banco
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    Set objContext = Nothing
    If adoCon Is Nothing Then   'Esta cerrada y es de VB
        Err.Raise Err.Number & " ", "dat_Contabilidad.Update.ContaPoliza", Err.Description
    Else        'Esta abierta
        If adoCon.Errors.Count = 0 Then     'Esta abierta y es de VB
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise Err.Number & " ", "dat_Contabilidad.Update.ContaPoliza",

Err.Description
            Exit Sub
        Else        'Esta abierta y es de DB
            For Each adoErr In adoCon.Errors
                str_ErrDesc = str_ErrDesc & " " & adoErr.Description
            Next
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise 10001, "dat_Contabilidad.Update.ContaPoliza", str_ErrDesc
        End If
    End If
End Sub

Dat_Contabilidad.Delete
Public Sub ContaCtaContable(ByVal ContaCta_Id As String)
' Autor:
' Fecha:
' Parametros:
'       ContaCta_Id = Id de la Cuenta
' Objetivo:
'       Eliminar físicamente el registro de la base de datos

Dim adoCon As ADODB.Connection
Dim adoErr As ADODB.Error
Dim objContext As MTxAS.ObjectContext
```

```
On Error GoTo errHandler

    Set objContext = GetObjectContext

    'Prepara la sentencia
    str_SQL = "Delete From ContaCtaContable Where " & _
    "ContaCta_Id='" & ContaCta_Id & "'"

    Set adoCon = CreateObject("ADODB.Connection")
    adoCon.ConnectionString = adoConString
    adoCon.Open
    adoCon.Execute str_SQL
    adoCon.Close
    Set adoCon = Nothing

    'Acepta la transacción
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    Set objContext = Nothing
    If adoCon Is Nothing Then    'Esta cerrada y es de VB
        Err.Raise Err.Number & " ", "dat_Contabilidad.Delete.ContaCtaContable",

Err.Description
    Else          'Esta abierta
        If adoCon.Errors.Count = 0 Then      'Esta abierta y es de VB
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise Err.Number & " ", "dat_Contabilidad.Delete.ContaCtaContable",

Err.Description
            Exit Sub
        Else         'Esta abierta y es de DB
            For Each adoErr In adoCon.Errors
                str_ErrDesc = str_ErrDesc & " " & adoErr.Description
            Next
            adoCon.Close
            Set adoCon = Nothing
            Err.Raise 10001, "dat_Contabilidad.Delete.ContaCtaContable", str_ErrDesc
        End If
    End If
End Sub
```

Listado 3. Objetos Emisarios

Bus_Contabilidad.Insert

```
Public Function ContaCtaContable(ByVal SysUser_Id As String, ByVal Cia_Id As String, _
    ByVal Reg_Id As String, ByVal Pra_Id As String, ByVal ContaCta_Id As String, _
    ByVal ContaCta_Desc As String, ByRef Reg_Ids As Variant, ByVal ContaHC_Bal01 As String)
' Autor:
' Fecha:
' Parametros:
'       Cia_Id                  =   Id de la Compañia
'       Reg_Id                  =   Id de la Region
'       Pra_Id                  =   Id del Fraccionamiento
'       ContaCta_Id             =   Id de la Cuenta
'       ContaCta_Desc           =   Descripcion de la cuenta
'       Reg_Ids()               =   Id de las regiones a la que esta asignada la cuenta
'       ContaHC_Bal01           =   Saldo inicial de la cuenta
'       SysUser_Id              =   Id del usuario
'
' Objetivo:
'       Insertar físicamente el registro en la base de datos

Dim objNegocios As util_Contabilidad.Negocios
Dim objInsert As dat_Contabilidad.Insert
Dim objUtilerias As Util_Contabilidad.Utilerias
Dim str_Fecha As String
Dim int_AñoIni As Integer
Dim int_AñoFin As Integer
Dim int_Cont As Integer
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext


    Set objNegocios = objContext.CreateInstance("util_Contabilidad.Negocios")
    'Valida que el usuario pueda ejecutar esta accion
    objNegocios.UsuarioTienePermiso SysUser_Id, ALTA_CUENTAS, Cia_Id, Reg_Id, Pra_Id

    'Elimina los caracteres de "-" de la cadena
    Set objUtilerias = objContext.CreateInstance("Util_Contabilidad.Utilerias")
     ContaCta_Id = objUtilerias.QuitaCar(ContaCta_Id, "-")

    'Valida que la cuenta sea de la compañia
    'objNegocios.EsCtadeCia Cia_Id, ContaCta_Id

    'Valida que el parent de la cuenta pueda ser acumulativa
    If Not objNegocios.CtaParentValida(ContaCta_Id) Then Err.Raise 10,
```

```
"bus_Contabilidad.Insert.ContaCtaContable", "La cuenta " & ContaCta_Id & " no tiene una

estructura correcta."

    'Obtiene la fecha del servidor
    str_Fecha = objUtilerias.FechaAct
    Set objUtilerias = Nothing

    'Inserta la cuenta
    Set objInsert = objContext.CreateInstance("dat_Contabilidad.Insert")
    objInsert.ContaCtaContable ContaCta_Id, ContaCta_Desc, "D", "1", str_Fecha, str_Fecha,

SysUser_Id


    'Define la categoria de la cuenta y de su parent
    objNegocios.DfneCategCtaParent ContaCta_Id, "A"

    'Obtiene el año inicial y final de la contabilidad
    int_AñoIni = objNegocios.ObtenAñoInicial(Cia_Id)
    int_AñoFin = objNegocios.ObtenAñoFinal(Cia_Id)

    'Inserta en ContaHistorialCtaHist los años anteriores
    For int_Cont = int_AñoIni To int_AñoFin - 1
        objInsert.ContaHistorialCta ContaCta_Id, CStr(int_Cont), ContaHC_Bal01, 0, 0,

ContaHC_Bal01, 0, 0, _
        ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0,

ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, _
        ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0,

ContaHC_Bal01
    Next

    'Inserta en ContaHistorialCta el registro correspondiente al año actual
    objInsert.ContaHistorialCta ContaCta_Id, CStr(int_AñoFin), ContaHC_Bal01, 0, 0,

ContaHC_Bal01, 0, 0, _
    ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0,

ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, _
    ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0, ContaHC_Bal01, 0, 0,

ContaHC_Bal01

    'Inserta en la tabla de ContaCtaContableRegion las foraneas a las que pertenece la

cuenta
    For int_Cont = 1 To UBound(Reg_Ids, 2)
        objInsert.ContaCtaContableRegion ContaCta_Id, Reg_Ids(1, int_Cont), 1
    Next
    Set objInsert = Nothing
    Set objNegocios = Nothing

    'Acepta la transacción
    objContext.SetComplete
    Set objContext = Nothing

Exit Function
errHandler:
    objContext.SetAbort
    Set objContext = Nothing

    If Not objNegocios Is Nothing Then Set objNegocios = Nothing
    If Not objInsert Is Nothing Then Set objInsert = Nothing
    If Not objUtilerias Is Nothing Then Set objUtilerias = Nothing
    Err.Raise Err.Number & " ", "dat_Contabilidad.Insert.ContaCtaContable", Err.Description
End Function


Public Sub Poliza(ByVal SysUser_Id As String, ByVal Cia_Id As String, ByVal Reg_Id As

String, _
    ByVal Fra_Id As String, ByVal TipoPoliza_Id As String, ByVal Periodo_Id As String, _
    ByVal ContaPol_Concepto As String, ByVal ContaPol_FechaCon As String, ByRef arr_Detalle

As Variant, _
    ByVal RevisaExistencia As String)
' Autor:
' Fecha:
' Parametros:
'       Cia_Id                  =   Id de la Comañia
'       Reg_Id                  =   Id de la Foranea
'       Fra_Id                  =   Id del Fraccionamiento
'       TipoPoliza_Id           =   Tipo de Poliza (IG,EG,DR,CP,TE,etc.)
'       Periodo_Id              =   Periodo contable
'       ContaPol_Concepto       =   Concepto de la Poliza
'       ContaPol_FechaCon       =   Fecha de contabilización
'       arr_Detalle             =   Contiene el deatalle de movimientos
'       RevisaExistencia        =   Define si se revisara la existencia de la poliza en la

BD
'       SysUser_Id              =   Id del usuario
'
```

```
' Objetivo:
'       Inserta la poliza contable y hace la afectación necesaria

Dim int_Intentos As Integer
Dim objNegocios As util_Contabilidad.Negocios
Dim objUtilerias As util_Contabilidad.Utilerias
Dim str_Fecha As String
Dim cur_ContaPol_TotalPol As Currency
Dim lng_ContaPol_Id As Long
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
Intenta:
    Set objContext = GetObjectContext

    Set objNegocios = objContext.CreateInstance("util_Contabilidad.Negocios")
    'Valida que el usuario pueda ejecutar esta accion
    objNegocios.UsuarioTienePermiso SysUser_Id, ALTA_POLIZAS, Cia_Id, Reg_Id, Fra_Id

    'Valida la existencia en la BD
    If RevisaExistencia = "TRUE" Then
        objNegocios.ExisteDocumentoContable Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id,

Periodo_Id, _
            ContaPol_Concepto, ContaPol_FechaCon, arr_Detalle, SysUser_Id
    End If

    'Obtiene el total de la poliza y verifica la integridad del datalle
    cur_ContaPol_TotalPol = objNegocios.ObtenDetalleCuadrado(arr_Detalle)

    'Obtiene el consecutivo de la poliza
    lng_ContaPol_Id = objNegocios.ObtenNumeroPoliza(Cia_Id, Reg_Id, Fra_Id, Periodo_Id,

TipoPoliza_Id)

    'Obtiene la fecha del servidor
    Set objUtilerias = objContext.CreateInstance("Util_Contabilidad.Utilerias")
    str_Fecha = objUtilerias.FechaAct
    Set objUtilerias = Nothing

    'Inserta en ContaPoliza
    objNegocios.InsertContaPoliza Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, _
        CStr(lng_ContaPol_Id), ContaPol_Concepto, ContaPol_FechaCon, str_Fecha, str_Fecha, _
        cur_ContaPol_TotalPol, BALANCEADO, CONTABILIDAD, SysUser_Id
    'Inserta y contabiliza el detalle
    objNegocios.InsertDetalle Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, _
        CStr(lng_ContaPol_Id), arr_Detalle

    Set objNegocios = Nothing

    'Acepta la transacción
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    If Not objContext Is Nothing Then Set objContext = Nothing
    If Not objNegocios Is Nothing Then Set objNegocios = Nothing
    If Not objUtilerias Is Nothing Then Set objUtilerias = Nothing
    If Err.Number = 2627 Then
        int_Intentos = int_Intentos + 1
        If int_Intentos < 50 Then GoTo Intenta
    Else
        Err.Raise Err.Number & " ", "bus_Contabilidad.Insert.Poliza", Err.Description
    End If
End Sub

Bus_Contabilidad.Update


Public Function ContaCtaContable(ByVal SysUser_Id As String, ByVal Cia_Id As String, _
    ByVal Reg_Id As String, ByVal Fra_Id As String, ByVal ContaCta_Id As String, _
    Optional ByVal ContaCta_Desc As String, Optional ByVal ContaHC_Bal01 As String, _
    Optional ByVal ContaCta_Activa As String)

' Autor:
' Fecha:
' Parametros:
'       Cia_Id              =   Id de la Compañia
'       Reg_Id              =   Id de la Region
'       Fra_Id              =   Id del Fraccionamiento
'       ContaCta_Id         =   Id de la Cuenta
'       ContaCta_Desc       =   Descripcion de la cuenta
'       ContaHC_Bal01       =   Actualización del Saldo inicial de la cuenta
'       ContaCta_Activa     =   Status de la cuenta
'       SysUser_Id          =   Id del usuario
'
' Objetivo:
'       Actualiza físicamente el registro en la base de datos

Dim objUpdate As dat_Contabilidad.Update
Dim objNegocios As util_Contabilidad.Negocios
Dim objUtilerias As util_Contabilidad.Utilerias
Dim int_AñoIni As Integer
Dim int_AñoFin As Integer
```

```
Dim str_Fecha As String
Dim int_Cont As Integer
Dim cur_ContaHC_Bal01 As Currency
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext

    Set objNegocios = objContext.CreateInstance("util_Contabilidad.Negocios")
    'Valida que el usuario pueda ejecutar esta accion
    objNegocios.UsuarioTienePermiso SysUser_Id, ACTUALIZA_CUENTAS, Cia_Id, Reg_Id, Fra_Id

    'Elimina los caracteres de "-" de la cadena
    Set objUtilerias = objContext.CreateInstance("Util_Contabilidad.Utilerias")
    ContaCta_Id = objUtilerias.QuitaCar(ContaCta_Id, "-")

    'Valida que la cuenta sea de la compañia
    objNegocios.EsCtadeCia Cia_Id, ContaCta_Id

    'Obtiene la fecha del servidor
    str_Fecha = objUtilerias.FechaAct
    Set objUtilerias = Nothing

    Set objUpdate = objContext.CreateInstance("dat_Contabilidad.Update")
    'Codigo para el caso que el usuario elimine logicamente la cuenta
    If ContaCta_Activa = "0" Then
        'Borra la cuenta logicamente
        objUpdate.ContaCtaContable SysUser_Id:=SysUser_Id, ContaCta_Id:=ContaCta_Id, ContaCta_FechaCambio:=str_Fecha,
ContaCta_Activa:=ContaCta_Activa

        'Desasigna las foraneas a la cuenta fisicamente
        objUpdate.ContaCtaContableRegion ContaCta_Id, "%", 0
    Else
        'Actualiza la cuenta
        objUpdate.ContaCtaContable SysUser_Id:=SysUser_Id, ContaCta_Id:=ContaCta_Id, ContaCta_FechaCambio:=str_Fecha,
ContaCta_Desc:=ContaCta_Desc, ContaCta_Activa:=ContaCta_Activa

        'Asigna las foraneas a la cuenta fisicamente
        objUpdate.ContaCtaContableRegion ContaCta_Id, "%", ContaCta_Activa

    End If

    'Obtiene el año inicial y final de la contabilidad
    int_AñoIni = objNegocios.ObtenAñoInicial(Cia_Id)
    int_AñoFin = objNegocios.ObtenAñoFinal(Cia_Id)

    'Obtiene el saldo inicial de la cuenta
    cur_ContaHC_Bal01 = objNegocios.ObtenSaldoIni(Cia_Id, ContaCta_Id)

    'Define cual es el incremento o decremnto en el saldo inicial
    'La formula para obtener el incremento o decremento en el saldo es:
    'SaldoNuevo-SaldoActual=Incremento o Decremento en el saldo
    'Este es el saldo que tiene como argumento esta funcion
    'Pj. 120-100=20, se incremento el saldo en 20
    'Pj, 80-100=-20, se decrementa el saldo en 20
    cur_ContaHC_Bal01 = CCur(ContaHC_Bal01) - cur_ContaHC_Bal01
    'Ahora cur_ContaHC_Bal01 tiene el incremento o decremento en el saldo

    'Actualiza en ContaHistorialCtaHist el saldo de los años anteriores
    For int_Cont = int_AñoIni To int_AñoFin - 1
        objUpdate.ContaHistorialCtaHistSaldos ContaCta_Id, CStr(int_Cont), cur_ContaHC_Bal01
    Next

    'Actualiza el año en cuestion
    objUpdate.ContaHistorialCtaSaldos ContaCta_Id, CStr(cur_ContaHC_Bal01)

    Set objNegocios = Nothing
    Set objUpdate = Nothing


    'Acepta la transacción
    objContext.SetComplete
    Set objContext = Nothing

Exit Function
errHandler:
    objContext.SetAbort
    Set objContext = Nothing

    If Not objNegocios Is Nothing Then Set objNegocios = Nothing
    If Not objUpdate Is Nothing Then Set objUpdate = Nothing
    If Not objUtilerias Is Nothing Then Set objUtilerias = Nothing
    Err.Raise Err.Number & " ", "dat_Contabilidad.Update.ContaCtaContable", Err.Description
End Function


Public Sub Poliza(ByVal SysUser_Id As String, ByVal Cia_Id As String, ByVal Reg_Id As String, _
    ByVal Fra_Id As String, ByVal TipoPoliza_Id As String, ByVal Periodo_Id As String, ByVal ContaPol_Id As String, _
    ByVal ContaPol_Concepto As String, ByVal ContaPol_FechaCon As String, ByRef arr_Detalle As Variant)
' Autor:
' Fecha:
' Parametros:
'       Cia_Id              =   Id de la Comañia
'       Reg_Id              =   Id de la Foranea
'       Fra_Id              =   Id del Fraccionamiento
'       TipoPoliza_Id       =   Tipo de Poliza (IG,EG,DR,CP,TE,etc.)
```

```
'       Periodo_Id              =   Periodo contable
'       ContaPol_Id             =   Id de la Poliza
'       ContaPol_Concepto       =   Concepto de la Poliza
'       ContaPol_FechaCon       =   Fecha de contabilización
'       arr_Detalle             =   Contiene el deatalle de movimientos
'       SysUser_Id              =   Id del usuario
'
' Objetivo:
'       Actualiza la poliza contable y hace la afectación necesaria

Dim objNegocios As util_Contabilidad.Negocios
Dim objUtilerias As util_Contabilidad.Utilerias
Dim str_Fecha As String
Dim cur_ContaPol_TotalPol As Currency
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext

    Set objNegocios = objContext.CreateInstance("util_Contabilidad.Negocios")
    'Valida que el usuario pueda ejecutar esta accion
    objNegocios.UsuarioTienePermiso SysUser_Id, ACTUALIZA_POLIZAS, Cia_Id, Reg_Id, Fra_Id

    'Valida que la poliza pueda ser actualizada
    objNegocios.PuedeModificarDocto Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, _
        ContaPol_Id

    'Obtiene el total de la poliza y verifica la integridad del datalle
    cur_ContaPol_TotalPol = objNegocios.ObtenDetalleCuadrado(arr_Detalle)

    'Obtiene la fecha del servidor
    Set objUtilerias = objContext.CreateInstance("Util_Contabilidad.Utilerias")
    str_Fecha = objUtilerias.FechaAct
    Set objUtilerias = Nothing

    'Decontabiliza la poliza
    objNegocios.DeContabilizaPoliza Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, _
        ContaPol_Id

    'Actualiza el encabezado en ContaPoliza
    objNegocios.UpdateContaPoliza Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, _
        ContaPol_Id, ContaPol_Concepto, ContaPol_FechaCon, str_Fecha, cur_ContaPol_TotalPol, _
        BALANCEADO, SysUser_Id

    'Borra el detalle
    objNegocios.DeleteDetalle Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, ContaPol_Id

    'Inserta y contabiliza el detalle
    objNegocios.InsertDetalle Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, _
        ContaPol_Id, arr_Detalle

    Set objNegocios = Nothing

    'Acepta la transacción
    objContext.SetComplete
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    If Not objContext Is Nothing Then Set objContext = Nothing
    If Not objNegocios Is Nothing Then Set objNegocios = Nothing
    If Not objUtilerias Is Nothing Then Set objUtilerias = Nothing
    Err.Raise Err.Number & " ", "bus_Contabilidad.Update.Poliza", Err.Description
End Sub

BusContabilidad.Delete
Public Function ContaCtaContable(ByVal SysUser_Id As String, ByVal Cia_Id As String, _
    ByVal Reg_Id As String, ByVal Fra_Id As String, ByVal ContaCta_Id As String)
' Autor:
' Fecha:
' Parametros:
'       Cia_Id              =   Id de la Compañia
'       ContaCta_Id         =   Id de la Cuenta
'       SysUser_Id          =   Id del usuario
'
' Objetivo:
'       Borra físicamente el registro en la base de datos
Dim objUtilerias As util_Contabilidad.Utilerias
Dim objNegocios As util_Contabilidad.Negocios
Dim objDelete As dat_Contabilidad.Delete
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
    Set objContext = GetObjectContext

    Set objNegocios = objContext.CreateInstance("util_Contabilidad.Negocios")
    'Valida que el usuario pueda ejecutar esta accion
    objNegocios.UsuarioTienePermiso SysUser_Id, BAJA_CUENTAS, Cia_Id, Reg_Id, Fra_Id

    'Elimina los caracteres de "-" de la cadena
    Set objUtilerias = objContext.CreateInstance("Util_Contabilidad.Utilerias")
    ContaCta_Id = objUtilerias.QuitaCar(ContaCta_Id, "-")
    Set objUtilerias = Nothing

    'Valida que la cuenta sea de la compañia
    objNegocios.EsCtadeCia Cia_Id, ContaCta_Id
```

```
    'Valida que la cuenta no tenga movimientos
    If objNegocios.TieneMovimientos(ContaCta_Id) Then Err.Raise 10,

"bus_Contabilidad.Delete.ContaCtaContable", "La cuenta " & ContaCta_Id & " tiene movimientos

y no puede ser borrada."

    'Pone como cuenta detalle la cuenta parent
    objNegocios.DfneCategCtaParent ContaCta_Id, "D"
    Set objNegocios = Nothing

    Set objDelete = objContext.CreateInstance("dat_Contabilidad.Delete")

    'Borra la cuenta de ContaHistorialCta
    objDelete.ContaHistorialCta ContaCta_Id

    'Borra la cuenta de ContaHistorialCtaHist
    objDelete.ContaHistorialCtaHist ContaCta_Id

    'Borra de la tabla ContaCtaContableRegion
    objDelete.ContaCtaContableRegion ContaCta_Id

    'Borra la cuenta
    objDelete.ContaCtaContable ContaCta_Id

    Set objDelete = Nothing

    'Acepta la transacción
    objContext.SetComplete
    Set objContext = Nothing

Exit Function
errHandler:
    objContext.SetAbort
    Set objContext = Nothing
    If Not objNegocios Is Nothing Then Set objNegocios = Nothing
    If Not objDelete Is Nothing Then Set objDelete = Nothing
    Err.Raise Err.Number & " ", "dat_Contabilidad.Delete.ContaCtaContable", Err.Description
End Function

Public Sub Poliza(ByVal SysUser_Id As String, ByVal Cia_Id As String, ByVal Reg_Id As

String, _
    ByVal Fra_Id As String, ByVal TipoPoliza_Id As String, ByVal Periodo_Id As String, _
    ByVal ContaPol_Id As String)
' Autor:
' Fecha:
' Parametros:
'       Cia_Id              =   Id de la Comañia
'       Reg_Id              =   Id de la Foranea
'       Fra_Id              =   Id del Fraccionamiento
'       TipoPoliza_Id       =   Tipo de Poliza (IG,EG,DR,CP,TE,etc.)
'       Periodo_Id          =   Periodo contable
'       ContaPol_Id         =   Id de lapoliza
'       SysUser_Id          =   Id del usuario
'
' Objetivo:
'       Borra logicamente la poliza

Dim int_Intentos As Integer
Dim objNegocios As util_Contabilidad.Negocios
Dim objUtilerias As util_Contabilidad.Utilerias
Dim str_Fecha As String
Dim objContext As MTxAS.ObjectContext
On Error GoTo errHandler
Intenta:
    Set objContext = GetObjectContext

    Set objNegocios = objContext.CreateInstance("util_Contabilidad.Negocios")
    'Valida que el usuario pueda ejecutar esta accion
    objNegocios.UsuarioTienePermiso SysUser_Id, BAJA_POLIZAS, Cia_Id, Reg_Id, Fra_Id

    'Valida que la poliza pueda ser borrada
    objNegocios.PuedeModificarDocto Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, _
        ContaPol_Id

    'Obtiene la fecha del servidor
    Set objUtilerias = objContext.CreateInstance("Util_Contabilidad.Utilerias")
    str_Fecha = objUtilerias.FechaAct
    Set objUtilerias = Nothing


    'Borra logicamente de  ContaPoliza
    objNegocios.DeleteContaPoliza Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, _
        ContaPol_Id, str_Fecha, SysUser_Id

    'Decontabiliza la poliza
    objNegocios.DeContabilizaPoliza Cia_Id, Reg_Id, Fra_Id, TipoPoliza_Id, Periodo_Id, _
        ContaPol_Id

    Set objNegocios = Nothing

    'Acepta la transacción
    objContext.SetComplete
```

```
    Set objContext = Nothing

Exit Sub
errHandler:
    objContext.SetAbort
    If Not objContext Is Nothing Then Set objContext = Nothing
    If Not objNegocios Is Nothing Then Set objNegocios = Nothing
    If Not objUtilerias Is Nothing Then Set objUtilerias = Nothing
    Err.Raise Err.Number & " ", "bus_Contabilidad.Insert.Poliza", Err.Description
End Sub
```

Listado 4. Página de registro, actualización y eliminación de Cuentas Contables

```
<%@ Language=VBScript %>
<SCRIPT id=DebugDirectives runat=server language=javascript>
// Set these to true to enable debugging or tracing
@set @debug=false
@set @trace=false
</SCRIPT>

<%
'Variables para desplegar los valores en la pagina
Dim varCboReg_Id      'Guarda el Valor del Combo de la Region
Dim varlstReg_Id      '
Dim varReg_Ids                          'Contiene las foraneas a las que pertenece esa cuenta

'Variables que contiene los valores extraidos de la BD
Dim varContaCta_Id                              'Clave de la Cuenta
Dim varFmtContaCta_Id                           'Clave de la Cuenta con formato
Dim varContaCta_Desc
Dim varContaCta_Categoria
Dim varContaCta_Activa
Dim varSaldoIni
Dim varDebe
Dim varHaber
Dim SaldoFin

'Variables para el manejo de la cuenta
Dim objInsert
Dim objUpdate
Dim objDelete
Dim varAction
Dim varDisplay    '

'Estas variables son ambito global
Dim varSysUser_Id
Dim varCia_Id
Dim varReg_Id         'Clave de la Region
Dim varFra_Id

'Asigno los valores a las variables
varCia_Id="0002"
varReg_Id="0001"
varFra_Id="0100"
varSysUser_Id="1"



'Obtiene la accion a ejecutar
varAction           = Request.Form("hdnAction")
varContaCta_Id      =Request.QueryString("ContaCta_Id")

'Verifica si la pagina se esta llamando con una cuenta especifica
if varContaCta_Id<>"" then
        On Error Resume Next
        'Formatea la cuenta
        Set objUtilerias            =Server.CreateObject("util_Contabilidad.Utilerias")
        varFmtContaCta_Id           =objUtilerias.FormatCuenta(varContaCta_Id)
        if Err.number  <> 0 then
                Set objUtilerias = nothing
                Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Cuentas/ContaCtaContable.asp"
        end if
        set objUtilerias= nothing

        'Prepara la cadena
        str_SQL="Select * from ContaCtaContable " & _
                        "Where ContaCta_Id='" & varContacta_Id & "'"
        'Realizo conexion y coloco los campos en las variables
        Set adoRS = Server.CreateObject("ADODB.Recordset")
        adoRS.Open  str_SQL,Application("ContaConn_ConnectionString")
        if adoRs.EOF and adoRs.BOF then
                adoRs.Close
                set adoRS = nothing
                Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=La cuenta " & varFmtContaCta_Id & " no existe
en el catalogo&URL=./Cuentas/ContaCtaContable.asp"
        end if

        varContaCta_Desc            =adoRs.Fields("ContaCta_Desc")
        varContaCta_Categoria       =adoRs.Fields("ContaCta_Categoria")
        varContaCta_Activa          =adoRs.Fields("ContaCta_Activa")
        varContaCta_FechaAlta       =adoRs.Fields("ContaCta_FechaAlta")
        varContaCta_SysUser_Id      =adoRs.Fields("SysUser_Id")
        adoRs.Close
        set adoRS = nothing
```

```
        'Formatea la categoria
        if Trim(varContaCta_Categoria)="A" then
                varContaCta_Categoria          ="CUENTA: ACUMULATIVA"
        else
                varContaCta_Categoria          ="CUENTA: DETALLE"
        end if
        'Formate si la cuenta esta activa
        if varContaCta_Activa then varContaCta_Activa      ="CHECKED"

        'Formatea la fecha
        varContaCta_FechaAlta=Clng(DateSerial(year(varContaCta_FechaAlta),Month(varContaCta_FechaAlta),Day(varContaCt
a_FechaAlta)))

        'Obtiene los saldos Inicial,Debe,Haber y SaldoFin
        Set objUtilerias               =Server.CreateObject("util_Contabilidad.Negocios")
        varSaldoIni                                    =objUtilerias.ObtenSaldoIni(varCia_Id, varContaCta_Id)
        varDebe                                        =objUtilerias.ObtenTotalDebe(varContaCta_Id)
        varHaber                                  =objUtilerias.ObtenTotalHaber(varContaCta_Id)
        if Err.number  <> 0 then
                Set objUtilerias = nothing
                Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Cuentas/ContaCtaContable.asp"
        end if
        set objUtilerias= nothing
        varSaldoFin=varSaldoIni + varDebe + varHaber

        varDisplay="CONSULTA"
else

        select case varAction
        case "INSERTAR"
                On error Resume Next
                'Obtiene los valores enviados por el usuario
                varFmtContaCta_Id   =UCase(Request.Form("ctlContaCta_Id"))
                varhdnReg_Ids                  =Request.Form("hdnReg_Ids")
                varContaCta_Desc    =UCase(Request.Form("txtContaCta_Desc"))
                varContaHC_Bal01    =Request.Form("txtSaldoIni")

                'Obtiene el arreglo a partir de la cadena
                set objUtilerias=Server.CreateObject("Util_Contabilidad.Utilerias")
                varContaCta_Id      =objUtilerias.QuitaCar(varFmtContaCta_Id,"-")
                varReg_Ids                     =objUtilerias.GeneraArreglo(varhdnReg_Ids)
                if Err.number  <> 0 then
                        Set objUtilerias = nothing
                        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Cuentas/ContaCtaContable.asp"
                end if
                Set objUtilerias = nothing

                'Con los valores obtenidos por el usuario
        Set objInsert= Server.CreateObject("bus_Contabilidad.Insert")
                        objInsert.ContaCtaContable
varSysUser_Id,varCia_Id,varReg_Id,varFra_Id,varContaCta_Id,varContaCta_Desc, varReg_Ids,varContaHC_Bal01
                if Err.number  <> 0 then
                        Set objInsert = nothing
                        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Cuentas/ContaCtaContable.asp"
                end if
        set objInsert= Nothing
                Response.Redirect "ContaCtaContable.asp?ContaCta_Id=" & varContaCta_Id
          case "BORRAR"
                On error resume next
                'Obtiene los valores enviados por el usuario
                varFmtContaCta_Id   =UCase(Request.Form("ctlContaCta_Id"))

                'Quita el formato a la cuenta
                set objUtilerias=Server.CreateObject("Util_Contabilidad.Utilerias")
                varContaCta_Id      =objUtilerias.QuitaCar(varFmtContaCta_Id,"-")
                if Err.number  <> 0 then
                        Set objUtilerias = nothing
                        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Cuentas/ContaCtaContable.asp?ContaCta_Id=" & varContaCta_Id
                end if
                Set objUtilerias = nothing

                'Con los valores obtenidos por el usuario
        Set objDelete= Server.CreateObject("bus_Contabilidad.Delete")
                        objDelete.ContaCtaContable  varSysUser_Id,varCia_Id,varReg_Id,varFra_Id,varContaCta_Id
                if Err.number  <> 0 then
                        Set objDelete = nothing
                        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Cuentas/ContaCtaContable.asp?ContaCta_Id=" & varContaCta_Id
                end if
        set objDelete= Nothing
                Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=La cuenta ha sido borrada con
exito.&URL=./Cuentas/CatContaCtaContable.asp"

          case "ACTUALIZAR"
                On error resume next
                'Obtiene los valores enviados por el usuario
                varFmtContaCta_Id   =UCase(Request.Form("ctlContaCta_Id"))
                varContaCta_Desc    =UCase(Request.Form("txtContaCta_Desc"))
                varContaHC_Bal01    =Request.Form("txtSaldoIni")
                varContaCta_Activa  =Request.Form("hdnContaCta_Activa")
```

```
                    'Quita el formato a la cuenta
                    set objUtilerias=Server.CreateObject("Util_Contabilidad.Utilerias")
                    varContaCta_Id      =objUtilerias.QuitaCar(varFmtContaCta_Id,"-")
                    if Err.number  <> 0 then
                              Set objUtilerias = nothing
                              Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Cuentas/ContaCtaContable.asp"
                    end if
                    Set objUtilerias = nothing

                    'Con los valores obtenidos por el usuario
          Set objUpdate= Server.CreateObject("bus_Contabilidad.Update")
                              objUpdate.ContaCtaContable
var$ysUser_Id,varCia_Id,varReg_Id,varFra_Id,varContaCta_Id,varContaCta_Desc, varContaHC_Bal01,varContaCta_Activa
                    if Err.number  <> 0 then
                              Set objUpdate = nothing
                              Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Cuentas/ContaCtaContable.asp"
                    end if
          set objUpdate= Nothing
                    Response.Redirect "ContaCtaContable.asp?ContaCta_Id=" & varContaCta_Id
        end select
end if
%>

<html>
<head>
<META name=VI60_defaultClientScript content=VBScript>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<!--#Include file="../HighLowLight.asp"-->
<link href="../../Styles/styleContabilidad.css" rel="stylesheet" type="text/css">
</head>
<body>

<form Name="frmContaCtaContable" Action="ContaCtaContable.asp" Method="post">
<%Select case varDisplay%>
<%                  Case "DELETE"                   %>
    <table bgColor="#336699" border="1" cellPadding="1" cellSpacing="1" width="100%" background =""
borderColorLight="dodgerblue" style="WIDTH: 100%">
        <tr>
          <td>
            <p align="center"><font color="white">BAJA CUENTAS CONTABLES</font></p>
          </td>
        </tr>
          </table>
          <br>
          <br>
          <hr>
          <!-- >              Esta tabla es independiente de la de arriba y contiene un registro de dos columnas
          en cada columna esta otra tabla          <-->

          <table border="0" cellpadding="1" cellspacing="1" width="100%">
          <tr>
          <td>
                    <table border="0" cellpadding="1" cellspacing="1" width="100%">
                              <tr>
                                        <td>CUENTA:
                                        </td>
                              </tr>
                              <tr>
                                        <td>
            <OBJECT classid=clsid:C932BA85-4374-101B-A56C-00AA003668DC height=15
            id=ctlContaCta_Id name=ctlContaCta_Id
            style="HEIGHT: 15px; LEFT: 0px; TOP: 0px; WIDTH: 180px" width=180
            VIEWASTEXT><PARAM NAME="_ExtentX" VALUE="5292"><PARAM NAME="_ExtentY" VALUE="582">
            <PARAM NAME="_Version" VALUE="393216"><PARAM NAME="BorderStyle" VALUE="0">
            <PARAM NAME="ClipMode" VALUE="0"><PARAM NAME="MousePointer" VALUE="0">
            <PARAM NAME="Appearance" VALUE="1"><PARAM NAME="BackColor" VALUE="-2147483643">
            <PARAM NAME="ForeColor" VALUE="-2147483640"><PARAM NAME="PromptInclude" VALUE="-1">
            <PARAM NAME="AllowPrompt" VALUE="0"><PARAM NAME="AutoTab" VALUE="0">
            <PARAM NAME="HideSelection" VALUE="-1"><PARAM NAME="Enabled" VALUE="-1">
            <PARAM NAME="MaxLength" VALUE="25"><PARAM NAME="OLEDragMode" VALUE="0">
            <PARAM NAME="OLEDropMode" VALUE="0"><PARAM NAME="Mask" VALUE="&amp;amp;&amp;amp;&amp;amp;-&amp;
&amp;amp;&amp;amp;&amp;amp;-&amp;amp;&amp;amp;&amp;amp;-&amp;amp;&amp;amp;&amp;amp;-&amp;amp;&amp;amp;&amp;amp;">
            <PARAM NAME="PromptChar" VALUE="_"></OBJECT>
                                        </td>
                              </tr>
                              <tr>
                                        <td>DESCRIPCION:
                                        </td>
                              </tr>
                              <tr>
                                        <td>
                                                  <input id="txtContaCta_Desc" name="txtContaCta_Desc" style="HEIGHT:
22px; WIDTH: 582px" maxlength=150>
                                        </td>
                              </tr>
                    </table>
          </td>
          <td valign=Top>
                    <table border="0" cellpadding="1" cellspacing="1" width="100%">
                              <tr>
                                        <td align="right">
```

```
                                        <object classid="clsid:20DD1B9E-87C4-11D1-8BE3-0000F8754DA1"
codebase="../../../../../cab/Mscomct2.ocx" height="26"
                                        id="ctlFecha" name="ctlFecha" style="HEIGHT: 26px; LEFT: 0px; TOP: 0px; WIDTH:
100px" width="100" VIEWASTEXT>
                                        <param NAME="_ExtentX" VALUE="2646">
                                        <param NAME="_ExtentY" VALUE="688">
                                        <param NAME="_Version" VALUE="393216">
                                        <param NAME="MousePointer" VALUE="0">
                                        <param NAME="Enabled" VALUE="1">
                                        <param NAME="OLEDropMode" VALUE="0">
                                        <param NAME="CalendarBackColor" VALUE="-2147483643">
                                        <param NAME="CalendarForeColor" VALUE="-2147483630">
                                        <param NAME="CalendarTitleBackColor" VALUE="-2147483633">
                                        <param NAME="CalendarTitleForeColor" VALUE="-2147483630">
                                        <param NAME="CalendarTrailingForeColor" VALUE="-2147483631">
                                        <param NAME="CheckBox" VALUE="0">
                                        <param NAME="CustomFormat" VALUE="dd/MM/yyyy">
                                        <param NAME="DateIsNull" VALUE="0">
                                        <param NAME="Format" VALUE="662831107">
                                        <param NAME="UpDown" VALUE="0">
                                        <param NAME="CurrentDate" VALUE="<%=varContaCta_FechaAlta%>">
                                        <param NAME="MaxDate" VALUE="<%=varContaCta_FechaAlta%>">
                                        <param NAME="MinDate" VALUE="<%=varContaCta_FechaAlta%>">
                                        </object>
                                        </td>
                                </tr>
                                <tr>
                                        <td align="right">
                                                <IMG height=20 id=btnInsertar name=btnInsertar onmouseout=LowLight()
onmouseover=HighLight() src="../../images/btnInsertar_Low.gif" style="CURSOR: hand" width=80>
                                        </td>
                                </tr>
                                <tr>
                                        <td align="right">
                                                <IMG height=20 id=btnActualizar name=btnActualizar
onmouseout=LowLight() onmouseover=HighLight() src="../../images/btnActualizar_Low.gif" style="CURSOR: hand" width=80>
                                        </td>
                                </tr>
                                <tr>
                                        <td align="right">
                                                <IMG height=20 id=btnBorrar name=btnBorrar onmouseout=LowLight()
onmouseover=HighLight() src="../../images/btnBorrar_Low.gif" style="CURSOR: hand" width=80>
                                        </td>
                                </tr>
                        </table>
                </td>
                </tr>
                </table>
                <br>
                <hr>
                <!-- >               Esta tabla es independiente de la de arriba        <-->
                <table border="0" width="100%">
                <tr>
                        <td align="left">SALDO INICIAL
                        </td>
                        <td align="left">CARGOS
                        </td>
                        <td align="left">ABONOS
                        </td>
                        <td align="left">SALDO FINAL
                        </td>
                </tr>
                <tr>
                        <td align="left"><input id="txtSaldoIni" name="txtSaldoIni" style="HEIGHT: 22px; WIDTH: 94px">
                        </td>
                        <td align="left"><input id="txtCargo" name="txtCargo" style="HEIGHT: 22px; WIDTH: 94px">
                        </td>
                        <td align="left"><input id="txtAbono" name="txtAbono" style="HEIGHT: 22px; WIDTH: 94px">
                        </td>
                        <td align="left"><input id="txtSaldoFin" name="txtSaldoFin" style="HEIGHT: 22px; WIDTH: 94px">
                        </td>
                </tr>
                </table>
                <br>
        <hr>
                <!-- >               Esta tabla es independiente de la de arriba        <-->
                <table border="0" width="100%">
                <tr>
                        <td align="left">REGIONES
                        </td>
                        <td align="left">REGIONES ASIGNADAS ACTUALMENTE
                        </td>
                </tr>
                <tr>
                        <td Valign="top">
                                <%
                                        set objCbo = CreateObject("util_Contabilidad.HTML")
                                                varcboReg_Id = objcbo.CboList
("cboReg_Id","15","300","Region","Reg_Id", "Reg_Id + ' ' + Reg_Desc"," Where Reg_Status = 1 Order by  Reg_Desc",
varReg_Id)
                                                varReg_Id= left(varcboReg_Id,4)
                                                Response.Write mid(varcboReg_Id,5)
                                        set objCbo = nothing
                                %>
                        </td>
```

```
                    <td align="left">
                        <%
                                set objlst = CreateObject("util_Contabilidad.HTML")
                                        varlstReg_Id = objlst.Listbox
("lstReg_Id",true,5,"100","300","Region","Reg_Id", "Reg_Id +' '+Reg_Desc","Where Reg_Status = 1 Order by
Reg_Desc",varlstRegion)
                                        varlstRegion= left(varlstReg_Id,4)
                                        Response.Write mid(varlstReg_Id,5)
                                set objCbo = nothing
                        %>
                    </td>
            </tr>
            </table>
<%                  Case "UPDATE"       %>
    <table bgColor="#336699" border="1" cellPadding="1" cellSpacing="1" width="100%" background =""
borderColorLight="dodgerblue" style="WIDTH: 100%">
        <tr>
          <td>
            <p align="center"><font color="white">ACTUALIZACION DE CUENTAS CONTABLES</font></p>
          </td>
        </tr>
          </table>
          <br>
          <br>
          <hr>
          <!-- >              Esta tabla es independiente de la de arriba y contiene un registro de dos columnas
          en cada columna esta otra tabla          <-->

          <table border="0" cellpadding="1" cellspacing="1" width="100%">
          <tr>
          <td>
                    <table border="0" cellpadding="1" cellspacing="1" width="100%">
                        <tr>
                                <td>CUENTA:
                                </td>
                        </tr>
                        <tr>
                                <td>
            <OBJECT classid=clsid:C932BA85-4374-101B-A56C-00AA003668DC height=15
            id=ctlContaCta_Id name=ctlContaCta_Id
            style="HEIGHT: 15px; LEFT: 0px; TOP: 0px; WIDTH: 180px" width=180
            VIEWASTEXT><PARAM NAME="_ExtentX" VALUE="5292"><PARAM NAME="_ExtentY" VALUE="582">
            <PARAM NAME="_Version" VALUE="393216"><PARAM NAME="BorderStyle" VALUE="0">
            <PARAM NAME="ClipMode" VALUE="0"><PARAM NAME="MousePointer" VALUE="0">
            <PARAM NAME="Appearance" VALUE="1"><PARAM NAME="BackColor" VALUE="-2147483643">
            <PARAM NAME="ForeColor" VALUE="-2147483640"><PARAM NAME="PromptInclude" VALUE="-1">
            <PARAM NAME="AllowPrompt" VALUE="0"><PARAM NAME="AutoTab" VALUE="0">
            <PARAM NAME="HideSelection" VALUE="-1"><PARAM NAME="Enabled" VALUE="-1">
            <PARAM NAME="MaxLength" VALUE="25"><PARAM NAME="OLEDragMode" VALUE="0">
            <PARAM NAME="OLEDropMode" VALUE="0"><PARAM NAME="Mask" VALUE="&amp;amp;&amp;amp;&amp;amp;-&amp;-
&amp;amp;&amp;amp;&amp;amp;-&amp;amp;&amp;amp;&amp;amp;-&amp;amp;&amp;amp;&amp;amp;-&amp;amp;&amp;amp;&amp;amp;-&amp;amp;&amp;amp;&amp;amp;">
            <PARAM NAME="PromptChar" VALUE="_"></OBJECT>
                                </td>
                        </tr>
                        <tr>
                                <td>DESCRIPCION:
                                </td>
                        </tr>
                        <tr>
                                <td>
                                        <input id="txtContaCta_Desc" name="txtContaCta_Desc" style="HEIGHT:
22px; WIDTH: 582px" maxlength=150>
                                </td>
                        </tr>
                    </table>
          </td>
          <td valign=Top>
                    <table border="0" cellpadding="1" cellspacing="1" width="100%">
                        <tr>
                                <td align="right">
            <OBJECT classid=clsid:20DD1B9E-87C4-11D1-8BE3-0000F8754DA1
            id=ctlFecha name=ctlFecha
            style="HEIGHT: 22px; LEFT: 0px; TOP: 0px; WIDTH: 100px" VIEWASTEXT><PARAM NAME="_ExtentX"
VALUE="2646"><PARAM NAME="_ExtentY" VALUE="582"><PARAM NAME="_Version" VALUE="393216"><PARAM NAME="MousePointer"
VALUE="0"><PARAM NAME="Enabled" VALUE="1"><PARAM NAME="OLEDropMode" VALUE="0"><PARAM NAME="CalendarBackColor"
VALUE="16777215"><PARAM NAME="CalendarForeColor" VALUE="12615680"><PARAM NAME="CalendarTitleBackColor"
VALUE="12632256"><PARAM NAME="CalendarTitleForeColor" VALUE="0"><PARAM NAME="CalendarTrailingForeColor" VALUE="-
2147483631"><PARAM NAME="CheckBox" VALUE="0"><PARAM NAME="CustomFormat" VALUE="dd/mm/yyyy"><PARAM NAME="DateIsNull"
VALUE="0"><PARAM NAME="Format" VALUE="662831107"><PARAM NAME="UpDown" VALUE="0"><PARAM NAME="CurrentDate"
VALUE="0"><PARAM NAME="MaxDate" VALUE="2958465"><PARAM NAME="MinDate" VALUE="-109205"></OBJECT>
                                </td>
                        </tr>
                        <tr>
                                <td align="right">
                                        <IMG height=20 id=btnInsertar name=btnInsertar onmouseout=LowLight()
onmouseover=HighLight() src="../../images/btnInsertar_Low.gif" style="CURSOR: hand" width=80>
                                </td>
                        </tr>
                        <tr>
                                <td align="right">
                                        <IMG height=20 id=btnActualizar name=btnActualizar
onmouseout=LowLight() onmouseover=HighLight() src="../../images/btnActualizar_Low.gif" style="CURSOR: hand" width=80>
                                </td>
                        </tr>
                        <tr>
```

```
                                        <td align="right">
                                                <IMG height=20 id=btnBorrar name=btnBorrar onmouseout=LowLight()
onmouseOver=HighLight() src="../../images/btnBorrar_Low.gif" style="CURSOR: hand" width=80>
                                        </td>
                                </tr>
                        </table>
                </td>
                </tr>
                </table>
                <br>
                <hr>
                <!-- >              Esta tabla es independiente de la de arriba       <-->
                <table border="0" width="100%">
                <tr>
                        <td align="left">SALDO INICIAL
                        </td>
                        <td align="left">CARGOS
                        </td>
                        <td align="left">ABONOS
                        </td>
                        <td align="left">SALDO FINAL
                        </td>
                </tr>
                <tr>
                        <td align="left"><input id="txtSaldoIni" name="txtSaldoIni" style="HEIGHT: 22px; WIDTH: 94px">
                        </td>
                        <td align="left"><input id="txtCargo" name="txtCargo" style="HEIGHT: 22px; WIDTH: 94px">
                        </td>
                        <td align="left"><input id="txtAbono" name="txtAbono" style="HEIGHT: 22px; WIDTH: 94px">
                        </td>
                        <td align="left"><input id="txtSaldoFin" name="txtSaldoFin" style="HEIGHT: 22px; WIDTH: 94px">
                        </td>
                </tr>
                </table>
                <br>
        <hr>
                <!-- >              Esta tabla es independiente de la de arriba       <-->
                <table border="0" width="100%">
                <tr>
                        <td align="left">REGIONES
                        </td>
                        <td align="left">REGIONES ASIGNADAS ACTUALMENTE
                        </td>
                </tr>
                <tr>
                        <td Valign="top">
                                <%
                                        set objCbo = CreateObject("util_Contabilidad.HTML")
                                                varcboReg_Id = objcbo.CboList
("cboReg_Id","15","300","Region","Reg_Id", "Reg_Id + ' ' + Reg_Desc"," Where Reg_Status = 1 Order by  Reg_Desc",
varReg_Id)
                                                varReg_Id= left(varcboReg_Id,4)
                                                Response.Write mid(varcboReg_Id,5)
                                        set objCbo = nothing
                                %>
                        </td>
                        <td align="left">
                                <%
                                        set objlst = CreateObject("util_Contabilidad.HTML")
                                                varlstReg_Id = objlst.Listbox
("lstReg_Id",true,5,"100","300","Region","Reg_Id", "Reg_Id +' '+Reg_Desc","Where Reg_Status = 1 Order by
Reg_Desc",varlstRegion)
                                                'varlstReg_Id = objlst.Listbox
("lstReg_Id",true,5,"100","300",varcboReg_Id,varcboReg_Id,varcboReg_Id)
                                                varlstRegion= left(varlstReg_Id,4)
                                                Response.Write mid(varlstReg_Id,5)
                                        set objCbo = nothing
                                %>
                        </td>
                </tr>
                </table>
<%              Case "CONSULTA"                 %>
    <table bgColor="#336699" border="1" cellPadding="1" cellSpacing="1" width="100%" background =""
borderColorLight="dodgerblue" style="WIDTH: 100%">
        <tr>
          <td>
            <p align="center"><font color="white">CONSULTA DE CUENTAS CONTABLES</font></p>
          </td>
        </tr>
          </table>
          <br>
          <br>
          <hr>
          <!-- >             Esta tabla es independiente de la de arriba y contiene un registro de dos columnas
          en cada columna esta otra tabla         <-->

          <table border="0" cellpadding="1" cellspacing="1" width="100%">
          <tr>
          <td>
                  <table border="0" cellpadding="1" cellspacing="1" width="100%">
                          <tr>
                                  <td>
                                          <%=varContaCta_Categoria%>
                                  </td>
                                  <td>
```

```
                                                <INPUT type="checkbox" id="chkContaCta_Activa"
name="chkContaCta_Activa" <%=varContaCta_Activa%>>ACTIVA
                                                </td>
                                    </tr>
                                    <tr>
                                                <td>CUENTA:
                                                </td>
                                    </tr>
                                    <tr>
                                                <td>
          <OBJECT classid=clsid:C932BA85-4374-101B-A56C-00AA003668DC height=15
          id=ctlContaCta_Id name=ctlContaCta_Id
          style="HEIGHT: 15px; LEFT: 0px; TOP: 0px; WIDTH: 180px" width=180
          VIEWASTEXT><PARAM NAME="_ExtentX" VALUE="5292"><PARAM NAME="_ExtentY" VALUE="582">
          <PARAM NAME="_Version" VALUE="393216"><PARAM NAME="BorderStyle" VALUE="0">
          <PARAM NAME="ClipMode" VALUE="0"><PARAM NAME="MousePointer" VALUE="0">
          <PARAM NAME="Appearance" VALUE="1"><PARAM NAME="BackColor" VALUE="-2147483643">
          <PARAM NAME="ForeColor" VALUE="-2147483640"><PARAM NAME="PromptInclude" VALUE="-1">
          <PARAM NAME="AllowPrompt" VALUE="0"><PARAM NAME="AutoTab" VALUE="0">
          <PARAM NAME="HideSelection" VALUE="-1"><PARAM NAME="Enabled" VALUE="-1">
          <PARAM NAME="MaxLength" VALUE="25"><PARAM NAME="OLEDragMode" VALUE="0">
          <PARAM NAME="OLEDropMode" VALUE="0"><PARAM NAME="Mask" VALUE="<%=varFmtContaCta_Id%>">
          <PARAM NAME="PromptChar" VALUE="_"></OBJECT>
                                                </td>
                                    </tr>
                                    <tr>
                                                <td>DESCRIPCION:
                                                </td>
                                    </tr>
                                    <tr>
                                                <td>
                                                            <input id="txtContaCta_Desc" name="txtContaCta_Desc"
value="<%=varContaCta_Desc%>" style="HEIGHT: 22px; WIDTH: 582px">
                                                </td>
                                    </tr>
                          </table>
          </td>
          <td valign=Top>
                          <table border="0" cellpadding="1" cellspacing="1" width="100%">
                                    <tr>
                                                <td align="right">
                                                <object classid="clsid:20DD1B9E-87C4-11D1-8BE3-0000F8754DA1"
codebase="../cab/Mscomct2.ocx" height="26"
                                                id="ctlFecha" name="ctlFecha" style="HEIGHT: 26px; LEFT: 0px; TOP: 0px; WIDTH:
100px" width="100" VIEWASTEXT>
                                                <param NAME="_ExtentX" VALUE="2646">
                                                <param NAME="_ExtentY" VALUE="688">
                                                <param NAME="_Version" VALUE="393216">
                                                <param NAME="MousePointer" VALUE="0">
                                                <param NAME="Enabled" VALUE="1">
                                                <param NAME="OLEDropMode" VALUE="0">
                                                <param NAME="CalendarBackColor" VALUE="-2147483643">
                                                <param NAME="CalendarForeColor" VALUE="-2147483630">
                                                <param NAME="CalendarTitleBackColor" VALUE="-2147483633">
                                                <param NAME="CalendarTitleForeColor" VALUE="-2147483630">
                                                <param NAME="CalendarTrailingForeColor" VALUE="-2147483631">
                                                <param NAME="CheckBox" VALUE="0">
                                                <param NAME="CustomFormat" VALUE="dd/MM/yyyy">
                                                <param NAME="DateIsNull" VALUE="0">
                                                <param NAME="Format" VALUE="662831107">
                                                <param NAME="UpDown" VALUE="0">
                                                <param NAME="CurrentDate" VALUE="<%=varContaCta_FechaAlta%>">
                                                <param NAME="MaxDate" VALUE="<%=varContaCta_FechaAlta%>">
                                                <param NAME="MinDate" VALUE="<%=varContaCta_FechaAlta%>">
                                                </object>
                                                </td>
                                    </tr>
                                    <tr>
                                                <td align="right">
                                                </td>
                                    </tr>
                                    <tr>
                                                <td align="right">
                                                            <IMG height=20 id=btnActualizar name=btnActualizar
onmouseout=LowLight() onmouseover=HighLight() src="../../images/btnActualizar_Low.gif" style="CURSOR: hand" width=80>
                                                </td>
                                    </tr>
                                    <tr>
                                                <td align="right">
                                                            <IMG height=20 id=btnBorrar name=btnBorrar onmouseout=LowLight()
onmouseover=HighLight() src="../../images/btnBorrar_Low.gif" style="CURSOR: hand" width=80>
                                                </td>
                                    </tr>
                          </table>
          </td>
          </tr>
          </table>
          <br>
          <hr>
          <!-- >                Esta tabla es independiente de la de arriba        <-->
          <table border="0" width="100%">
          <tr>
                    <td align="left">SALDO INICIAL
                    </td>
                    <td align="left">CARGOS
```

```
                    </td>
                    <td align="left">ABONOS
                    </td>
                    <td align="left">SALDO FINAL
                    </td>
          </tr>
          <tr>
                    <td align="left"><input id="txtSaldoIni" name="txtSaldoIni" style="HEIGHT: 22px; WIDTH: 94px"
          value="<%=varSaldoIni%>">
                    </td>
                    <td align="left"><input id="txtCargo" name="txtCargo" style="HEIGHT: 22px; WIDTH: 94px"
          value="<%=varDebe%>"                    readonly>
                    </td>
                    <td align="left"><input id="txtAbono" name="txtAbono" style="HEIGHT: 22px; WIDTH: 94px"
          value="<%=varHaber%>"                   readonly>
                    </td>
                    <td align="left"><input id="txtSaldoFin" name="txtSaldoFin" style="HEIGHT: 22px; WIDTH: 94px"
          value="<%=varSaldoFin%>"      readonly>
                    </td>
          </tr>
          </table>
          <br>
     <hr>
          <!-- >              Esta tabla es independiente de la de arriba        <-->
          <table border="0" width="100%">
          <tr>
                    <td align="left">REGIONES ASIGNADAS ACTUALMENTE
                    </td>
                    <td align="left">
                    </td>
          </tr>
          <tr>
                    <td Valign="top">
                              <%
                                        set objlst = CreateObject("util_Contabilidad.HTML")
                                                  varlstReg_Id = objlst.Listbox ("lstReg_Id
DISABLED",true,5,"100","300","Region,ContaCtaContableRegion","Region.Reg_Id", "Region.Reg_Id + ' ' + Region.Reg_Desc","
Where Region.Reg_Id=ContaCtaContableRegion.Reg_Id and ContaCtaContableRegion.ContaCta_Id='" & varContaCta_Id & "' and
ContaCtaContableRegion.Status = 1 Order by  Region.Reg_Desc")
                                                  if left(varlstReg_Id,1)<>"<" then
                                                            varlstReg_Id= mid(varlstReg_Id,5)
                                                  end if
                                                  Response.Write varlstReg_Id
                                        set objCbo = nothing
                              %>
                    </td>
                    <td align="left">
                    </td>
          </tr>
          </table>
<%case else 'INSERT%>
     <table bgColor="#3366999" border="1" cellPadding="1" cellSpacing="1" width="100%" background =""
borderColorLight="dodgerblue" style="WIDTH: 100%">
          <tr>
            <td>
              <p align="center"><font color="white">ALTA DE CUENTAS CONTABLES</font></p>
            </td>
          </tr>
            </table>
            <br>
            <br>
            <hr>
            <!-- >              Esta tabla es independiente de la de arriba y contiene un registro de dos columnas
            en cada columna esta otra tabla          <-->

            <table border="0" cellpadding="1" cellspacing="1" width="100%">
            <tr>
            <td>
                    <table border="0" cellpadding="1" cellspacing="1" width="100%">
                              <tr>
                                        <td>CUENTA:
                                        </td>
                              </tr>
                              <tr>
                                        <td>
              <OBJECT classid=clsid:C932BA85-4374-101B-A56C-00AA003668DC height=15
              id=ctlContaCta_Id name=ctlContaCta_Id
              style="HEIGHT: 15px; LEFT: 0px; TOP: 0px; WIDTH: 180px" width=180
              VIEWASTEXT><PARAM NAME="_ExtentX" VALUE="5292"><PARAM NAME="_ExtentY" VALUE="582">
              <PARAM NAME="_Version" VALUE="393216"><PARAM NAME="BorderStyle" VALUE="0">
              <PARAM NAME="ClipMode" VALUE="0"><PARAM NAME="MousePointer" VALUE="0">
              <PARAM NAME="Appearance" VALUE="1"><PARAM NAME="BackColor" VALUE="-2147483643">
              <PARAM NAME="ForeColor" VALUE="-2147483640"><PARAM NAME="PromptInclude" VALUE="-1">
              <PARAM NAME="AllowPrompt" VALUE="0"><PARAM NAME="AutoTab" VALUE="0">
              <PARAM NAME="HideSelection" VALUE="-1"><PARAM NAME="Enabled" VALUE="-1">
              <PARAM NAME="MaxLength" VALUE="25"><PARAM NAME="OLEDragMode" VALUE="0">
              <PARAM NAME="OLEDropMode" VALUE="0"><PARAM NAME="Mask" VALUE="&amp;amp;amp;amp;amp;-&amp;amp;
&amp;amp;amp;amp;amp;-&amp;amp;amp;amp;amp;-&amp;amp;amp;amp;amp;-&amp;amp;amp;amp;amp;">
              <PARAM NAME="PromptChar" VALUE="_"></OBJECT>
                                        </td>
                              </tr>
                              <tr>
                                        <td>DESCRIPCION:
                                        </td>
                              </tr>
```

```
                                <tr>
                                        <td>
                                                <input id="txtContaCta_Desc" name="txtContaCta_Desc" style="HEIGHT:
22px; WIDTH: 582px" maxlength=150 maxlength=150>
                                        </td>
                                </tr>
                        </table>
                </td>
                <td valign=Top>
                        <table border="0" cellpadding="1" cellspacing="1" width="100%">
                                <tr>
                                        <td align="right">
                                        </td>
                                </tr>
                                <tr>
                                        <td align="right">
                                                <IMG height=20 id=btnInsertar name=btnInsertar onmouseout=LowLight()
onmouseover=HighLight() src="../../images/btnInsertar_Low.gif" style="CURSOR: hand" width=80>
                                        </td>
                                </tr>
                                <tr>
                                        <td align="right">
                                        </td>
                                </tr>
                                <tr>
                                        <td align="right">
                                        </td>
                                </tr>
                        </table>
                </td>
                </tr>
                </table>
                <br>
                <hr>
                <!-- >               Esta tabla es independiente de la de arriba        <-->
                <table border="0" width="100%">
                <tr>
                        <td align="left">SALDO INICIAL
                        </td>
                        <td align="left">
                        </td>
                        <td align="left">
                        </td>
                        <td align="left">
                        </td>
                </tr>
                <tr>
                        <td align="left"><input id="txtSaldoIni" name="txtSaldoIni" style="HEIGHT: 22px; WIDTH: 94px"
value="$0.00">
                        </td>
                        <td align="left">
                        </td>
                        <td align="left">
                        </td>
                        <td align="left">
                        </td>
                </tr>
                </table>
                <br>
        <hr>
                <!-- >               Esta tabla es independiente de la de arriba        <-->
                <table border="0" width="100%">
                <tr>
                        <td align="left">REGIONES
                        </td>
                        <td align="left">REGIONES ASIGNADAS ACTUALMENTE
                        </td>
                </tr>
                <tr>
                        <td Valign="top">
                                <%
                                        set objCbo = CreateObject("util_Contabilidad.HTML")
                                                varcboReg_Id = objcbo.CboList
("cboReg_Id","15","300","Region","Reg_Id", "Reg_Id + ' ' + Reg_Desc"," Where Reg_Status = 1 Order by  Reg_Desc",
varReg_Id)
                                                varReg_Id= left(varcboReg_Id,4)
                                                Response.Write mid(varcboReg_Id,5)
                                        set objCbo = nothing
                                %>
                        </td>
                        <td align="left">
                                <%
                                        set objlst = CreateObject("util_Contabilidad.HTML")
                                                varlstReg_Id = objlst.Listbox
("lstReg_Id",true,5,"100","300","Region","Reg_Id", "Reg_Id +' '+Reg_Desc","Where Reg_Id='0000' and Reg_Status = 1 Order
by Reg_Desc",varlstRegion)
                                                'varlstReg_Id = objlst.Listbox
("lstReg_Id",true,5,"100","300",varcboReg_Id,varcboReg_Id,varcboReg_Id)
                                                varlstRegion= left(varlstReg_Id,4)
                                                Response.Write mid(varlstReg_Id,5)
                                        set objCbo = nothing
                                %>
                        </td>
                </tr>
                </table>
```

```
<%          End Select%>
            <input type="hidden" name="hdnAction" id="hdnAction" Value="<%=varAction%>">
            <input type="hidden" name="hdnContaCta_Activa" id="hdnContaCta_Activa">
            <input type="hidden" name="hdnReg_Ids" id="hdnReg_Ids">
</form>

</body>
</html>
<script ID="clientEventHandlersVBS" LANGUAGE="vbscript">

Sub btnInsertar_onclick()
            document.all("hdnAction").value="INSERTAR"
            'Obtiene los Ids que estan en el listbox de regiones seleccionadas para esa cuenta
            document.all("hdnReg_Ids").value=ObtenIdsDeListBox
            document.frmContaCtaContable.submit()
End Sub

Sub btnBorrar_onclick()
            document.all("hdnAction").value="BORRAR"
            document.frmContaCtaContable.submit()
End Sub

Sub btnActualizar_onclick()
            if document.all("chkContaCta_Activa").checked then
                        document.all("hdnContaCta_Activa").value="1"
            else
                        document.all("hdnContaCta_Activa").value="0"
            end if
            document.all("hdnAction").value="ACTUALIZAR"
            document.frmContaCtaContable.submit()
End Sub

Sub cboReg_Id_OnChange()
            Dim int_Cont
            if document.frmContaCtaContable.cboReg_Id.length=0 then
                        exit sub
            end if
            'Valido que la Region no este ya asociada
            For int_Cont = 0 to document.frmContaCtaContable.lstReg_Id.length - 1
                        if document.frmContaCtaContable.cboReg_Id.value =
document.frmContaCtaContable.lstReg_Id.options.item(int_Cont).value then
                                    msgbox "Esta foranea ya se encuentra asignada a la cuenta " &
Cstr(document.frmContaCtaContable.ctlContaCta_Id.Text), vbokOnly + vbExclamation,"Sintegrara2000"
                                    exit sub
                        End if
            Next
            'Adiciona la seleccion del combo al listbox
            Dim varOption
            set varOption=document.createElement("OPTION")
            varOption.text=document.frmContaCtaContable.cboReg_Id.Options(document.frmContaCtaContable.cboReg_Id.Selected
Index).text
            varOption.value=document.frmContaCtaContable.cboReg_Id.Options(document.frmContaCtaContable.cboReg_Id.Selecte
dIndex).value
            document.frmContaCtaContable.lstReg_Id.Options.Add varOption
End sub


sub lstReg_Id_OnDblClick()
            if document.frmContaCtaContable.lstReg_Id.length=0 then
                        exit sub
            end if
            'Adiciona la seleccion del combo al listbox
            document.frmContaCtaContable.lstReg_Id.Options.Remove  document.frmContaCtaContable.cboReg_Id.SelectedIndex
End sub

Function ObtenIdsDeListBox()
            Dim varNumItems
            Dim varIds

            for varNumItems=0 to  document.frmContaCtaContable.lstReg_Id.Length -1
                        document.frmContaCtaContable.lstReg_Id.SelectedIndex= varNumItems
                        varIds = varIds & "$$$&" & document.all("lstReg_Id").Value
            Next
            ObtenIdsDeListBox = "1," & Cstr(varNumItems) & varIds
end Function

Sub txtSaldoIni_onblur

End Sub

Sub txtSaldoIni_onfocus

End Sub

</script>
```

Listado 5. Página de registro, actualización y eliminación de Pólizas Contables

```
<%@ Language=VBScript %>

<%
'Variables para desplegar los valores en la pagina
Dim varCboReg_Id              'Guarda el Valor del Combo de la Region
Dim varCboFra_Id              'Guarda el valor del combo de la foranea
```

```
Dim varCia_Desc
Dim varPeriodo_Desc
Dim varSysUser_Nombre
Dim varContaPol_Concepto
Dim varContaPol_FechaAlta
Dim varContaPol_FechaIni
Dim varContaPol_FechaFin
Dim varContaPol_TotalPol

'Variables que contiene los valores extraidos de la BD
Dim vararr_Detalle


'Variables para el manejo de la poliza
Dim objInsert
Dim objUpdate
Dim objDelete
Dim varAction
Dim varDisplay

'Estas variables son ambito global
Dim varSysUser_Id
Dim varCia_Id
Dim varReg_Id          'Clave de la Region
Dim varFra_Id
Dim varPeriodo_Id
Dim varTipoPoliza_Id
Dim varContaPol_Id

'Asigno los valores a las variables
varCia_Id="0002"
varReg_Id="0001"
varFra_Id="0100"
varPeriodo_Id="200011"
varTipoPoliza="EG"
varSysUser_Id="1"



'Obtiene la accion a ejecutar
varAction                                   =Request.Form("hdnAction")

varContaPol_Id=Request.QueryString ("ContaPol_Id")

'Verifica si la pagina se esta llamando con una cuenta especifica
if varContaPol_Id<>"" then
        On Error Resume Next
        'Obtiene los valores
        varCia_Id                           =Request.QueryString("Cia_Id")
        varReg_Id                           =Request.QueryString("Reg_Id")
        varFra_Id                           =Request.QueryString("Fra_Id")
        varTipoPoliza_Id            =Request.QueryString("TipoPoliza_Id")
        varPeriodo_Id                       =Request.QueryString("Periodo_Id")

        'Obtiene las descripciones
        Set objUtilerias            =Server.CreateObject("util_Contabilidad.Negocios")
        varCia_Desc                                 =ObjUtilerias.ObtenNombreCia(varCia_Id)
        varSysUser_Nombre           =ObjUtilerias.ObtenNombreUsuario(varSysUser_Id)
        varContaPol_TotalPol
        =ObjUtilerias.ObtenTotalPoliza(varCia_Id,varReg_Id,varFra_Id,varTipoPoliza_Id,varPeriodo_Id, varContaPol_Id)
        vararr_Detalle                      =ObjUtilerias.ObtenDetalleString(varCia_Id,varReg_Id,varFra_Id,_

        varTipoPoliza_Id,varPeriodo_Id,varContaPol_Id)
        set objUtilerias= nothing
        varPeriodo_Desc = left(varPeriodo_Id,4) & " - " & Right(varPeriodo_Id,2)


        'Formatea la fecha
        varContaPol_FechaAlta=CDate(DateSerial(year(varContaPol_FechaAlta),Month(varContaPol_FechaAlta),Day(varContaPo
l_FechaAlta)))
        varDisplay="CONSULTA"
else

        select case varAction
        case "INSERTAR"
                On error Resume Next
                'Obtiene los valores enviados por el usuario
                varFra_Id                           =Request.Form("cboFra_Id")
                varTipoPoliza_Id            =Request.Form("cboTipoPoliza_Id")
                varContaPol_FechaAlta       =Request.Form("ctlFecha")
                varContaPol_Concepto        =UCase(Request.Form("txtContaPol_Concepto"))
                vararr_Detalle                      =Request.Form("hdnarr_Detalle")


                'Obtiene el arreglo a partir de la cadena
                set objUtilerias=Server.CreateObject("Util_Contabilidad.Utilerias")
                arr_Detalle                 =objUtilerias.GeneraArreglo(vararr_Detalle)
                if Err.number  <> 0 then
                        Set objUtilerias = nothing
                        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Polizas/ContaPoliza.asp"
                end if
                Set objUtilerias = nothing
```

```
                'Con los valores obtenidos por el usuario
        Set objInsert= Server.CreateObject("bus_Contabilidad.Insert")
                        varContaPol_Id = objInsert.Poliza
varSysUser_Id,varCia_Id,varReg_Id,varFra_Id,varTipoPoliza_Id,varPeriodo_Id, _

          varContaPol_Concepto,varContaPol_FechaAlta,arr_Detalle,"TRUE"
                . if Err.number  <> 0 then
                        . Set objInsert = nothing
                        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Polizas/ContaPoliza.asp"
                end if
        set objInsert= Nothing
                Response.Redirect "ContaPoliza.asp?Cia_Id=" & varCia_Id & "&Reg_Id=" & varReg_Id & _
                                                                        "&Fra_Id=" & varFra_Id & "&TipoPoliza_Id=" &
varTipoPoliza_Id & _
                                                                        "&Periodo_Id=" & varPeriodo_Id & "&ContaPol_Id="
& varContaPol_Id
          case "BORRAR"
                On error resume next
                'Obtiene los valores enviados por el usuario
                varFra_Id                               =Request.Form("cboFra_Id")
                varTipoPoliza_Id                =Request.Form("cboTipoPoliza_Id")
                varContaPol_FechaAlta           =Request.Form("ctlFecha")
                varContaPol_Concepto            =UCase(Request.Form("txtContaPol_Concepto"))


                'Con los valores obtenidos por el usuario
        Set objUpdate= Server.CreateObject("bus_Contabilidad.Update")
                        varContaPol_Id = objUpdate.Poliza
varSysUser_Id,varCia_Id,varReg_Id,varFra_Id,varTipoPoliza_Id,varPeriodo_Id, _
                                                                                varContaPol_Id
                if Err.number  <> 0 then
                        Set objInsert = nothing
                        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Polizas/ContaPoliza.asp"
                end if
        set objInsert= Nothing
                Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=Poliza eliminada con
exito&URL=./Polizas/ContaPoliza.asp"

          case "ACTUALIZAR"
                'Obtiene los valores enviados por el usuario
                varFra_Id                               =Request.Form("cboFra_Id")
                varTipoPoliza_Id                =Request.Form("cboTipoPoliza_Id")
                varContaPol_FechaAlta           =Request.Form("ctlFecha")
                varContaPol_Concepto            =UCase(Request.Form("txtContaPol_Concepto"))
                vararr_Detalle                          =Request.Form("hdnarr_Detalle")


                'Con los valores obtenidos por el usuario
        Set objUpdate= Server.CreateObject("bus_Contabilidad.Update")
                        varContaPol_Id = objUpdate.Poliza
varSysUser_Id,varCia_Id,varReg_Id,varFra_Id,varTipoPoliza_Id,varPeriodo_Id, _

          varContaPol_Id,varContaPol_Concepto,varContaPol_FechaAlta,arr_Detalle,"FALSE"

                if Err.number  <> 0 then
                        Set objInsert = nothing
                        Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=" & Err.Description  &
"&URL=./Polizas/ContaPoliza.asp"
                end if
        set objInsert= Nothing
                Response.Redirect "../InfoMsgContabilidad.asp?MENSAJE=Poliza eliminada con
exito&URL=./Polizas/ContaPoliza.asp"

          'El caso que se este visualizando la pagina por primera vez, entonces es un alta
          case else
          'Obtiene las descripciones
          Set objUtilerias              =Server.CreateObject("util_Contabilidad.Negocios")
          'varCia_Desc                                  =objUtilerias.ObtenNombreCia(varCia_Id)
          'varSysUser_Nombre            =ObjUtilerias.ObtenNombreUsuario(varSysUser_Id)
          set objUtilerias= nothing
          varPeriodo_Desc = left(varPeriodo_Id,4) & " - " & Right(varPeriodo_Id,2)

          'Formatea la fecha
          varContaPol_FechaAlta=Cing(DateSerial(year(varContaPol_FechaAlta),Month(varContaPol_FechaAlta),Day(varContaPo
l_FechaAlta)))

          end select
end if
%>


<html>
<head>
<META name=VI60_defaultClientScript content=VBScript>
<meta NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<link rel="stylesheet" type="text/css" href="../../Styles/styleContabilidad.css">
<!--#Include file="../HighLowLight.asp"-->
<OBJECT CLASSID="clsid:5220cb21-c88d-11cf-b347-00aa00a28331" id="Microsoft_Licensed_Class_Manager_1_0"1>
          <PARAM NAME="LPKPath" VALUE="../lpk/UcGrid.lpk">
</OBJECT>

</head>
<body>
```

```
<form NAME="frmContapoliza"  Action="ContaPoliza.asp" method=post >
<%Select Case varDisplay%>
<%                  Case "DELETE"                 %>
<%                  Case "UPDATE"                 %>
<%                  Case "CONSULTA"               %>
<table valign="Top" bgColor="#336699" border="1" cellPadding="1" cellSpacing="1" width="100%" background =""
borderColorLight="dodgerblue" style="WIDTH: 100%">
    <tr>
      <td>
        <p align="center"><font color="white">Consulta de Polizas</font></p>
      </td>
    </tr>
</table>
<!-- >                  Esta tabla es independiente de la de arriba y contiene un registro de dos columnas
          en cada columna esta otra tabla           <-->
<TABLE WIDTH="100%" BORDER=0 CELLSPACING=1 CELLPADDING=1>
          <TR>
                    <TD>Compañia:</TD>
                    <TD ColSpan=5>
                              <INPUT id=txtCia_Desc name=txtCia_Desc
                               Value="<%=varCia_Desc%>" readonly style="HEIGHT: 18px; LEFT: 0px; TOP: 0px; WIDTH:
580px" >
                    </TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD rowspan=2>
                              <TABLE WIDTH="100%" BORDER=0 CELLSPACING=1 CELLPADDING=1>
                                        <tr>
                                                  <td align="right">
                                                            <IMG height=20 id=btnActualizar name=btnActualizar
onmouseout=LowLight() onmouseover=HighLight() src="../../images/btnActualizar_Low.gif" style="CURSOR: hand" width=80>
                                                  </td>
                                        </tr>
                                        <tr>
                                                  <td align="right">
                                                            <IMG height=20 id=btnBorrar name=btnBorrar
onmouseout=LowLight() onmouseover=HighLight() src="../../images/btnBorrar_Low.gif" style="CURSOR: hand" width=80>
                                                  </td>
                                        </tr>
                              </TABLE>
                    </TD>
          </TR>
          <TR>
                    <TD>Foranea:</TD>
                    <TD>
                    <%
                              set objCbo = CreateObject("util_Contabilidad.HTML")
                                        varcboReg_Id = objcbo.CboList ("cboReg_Id
tabIndex=1","15","200","Region","Reg_Id","Reg_Id+' '+Reg_Desc","Where Reg_Status = 1 Order by  Reg_Desc", varReg_Id)
                                        if left(varcboReg_Id,1) <> "<" then
                                                  varReg_Id= left(varcboReg_Id,4)
                                                  Response.Write mid(varcboReg_Id,5)
                                        else
                                                  Response.Write varcboReg_Id
                                        end if
                              set objCbo = nothing
                    %>
                    </TD>
                    <TD>Fraccionamiento:</TD>
                    <TD>
                    <%
                              set objCbo = CreateObject("util_Contabilidad.HTML")
                                        varcboFra_Id = objcbo.CboList ("cboFra_Id
tabIndex=2","15","100","FraCia","Fra_Id","Fra_Id","Where Status = 1 Order by  Fra_Id", varFra_Id)
                                        if left(varcboFra_Id,1) <> "<" then
                                                  varFra_Id= left(varcboFra_Id,4)
                                                  Response.Write mid(varcboFra_Id,5)
                                        else
                                                  Response.Write varcboFra_Id
                                        end if
                              set objCbo = nothing
                    %>
                    </TD>
                    <TD>Fecha:</TD>
                    <TD>
      <OBJECT classid=clsid:20DD1B9E-87C4-11D1-8BE3-0000F8754DA1
       codebase="../cab/Msacmct2.ocx" id=ctlFecha name=ctlFecha tabIndex=3
      style="HEIGHT: 18px; LEFT: 0px; TOP: 0px; WIDTH: 100px" width=100
      VIEWASTEXT><PARAM NAME="_ExtentX" VALUE="2646"><PARAM NAME="_ExtentY" VALUE="476"><PARAM NAME="_Version"
VALUE="393216"><PARAM NAME="MousePointer" VALUE="0"><PARAM NAME="Enabled" VALUE="1"><PARAM NAME="OLEDropMode"
VALUE="0"><PARAM NAME="CalendarBackColor" VALUE="-2147483643"><PARAM NAME="CalendarForeColor" VALUE="-
2147483630"><PARAM NAME="CalendarTitleBackColor" VALUE="-2147483633"><PARAM NAME="CalendarTitleForeColor" VALUE="-
2147483630"><PARAM NAME="CalendarTrailingForeColor" VALUE="-2147483631"><PARAM NAME="CheckBox" VALUE="0"><PARAM
NAME="CustomFormat" VALUE="dd/MM/yyyy"><PARAM NAME="DateIsNull" VALUE="0"><PARAM NAME="Format" VALUE="662811107"><PARAM
NAME="UpDown" VALUE="0"><PARAM NAME="CurrentDate" VALUE="36494"><PARAM NAME="MaxDate" VALUE="2958465"><PARAM
NAME="MinDate" VALUE="-109205"></OBJECT>
                    </TD>
                    <TD></TD>
          </TR>
          <TR>
                    <TD>Periodo:</TD>
                    <TD>
                    <INPUT id=txtPeriodo_Id name=txtPeriodo_Id readonly
```

```
                    value="<%=varPeriodo_Desc%>"
                    style="HEIGHT: 18px; LEFT: 1px; TOP: 2px; WIDTH: 100px" >

                    </TD>
                    <TD>Documento:</TD>
                    <TD>
                    <%
                        set objCbo = CreateObject("util_Contabilidad.HTML")
                                    varcboTipoPoliza_Id = objcbo.CboList ("cboTipoPoliza_Id
tabIndex=4","15","100","TipoPoliza","TipoPoliza_Id","TipoPoliza_Id + ' ' + TipoPoliza_Desc","Where TipoPoliza_Status =
1 Order by TipoPoliza_Id", varTipoPoliza_Id)
                                    If left(varcboTipoPoliza_Id,1) <> "<" then
                                            varTipoPoliza_Id= left(varcboTipoPoliza_Id, 2)
                                            Response.Write
Mid(varcboTipoPoliza_Id,instr(varcboTipoPoliza_Id,"<"))
                                    else
                                            Response.Write varcboTipoPoliza_Id
                                    end if
                            set objCbo = nothing
                    %>
                    </TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
            </TR>
            <TR>
                    <TD>Concepto:</TD>
                    <TD ColSpan=5>
                            <INPUT id=txtContaPol_Concepto name=txtContaPol_Concepto
Value="<%=varContaPol_Concepto%>" style="HEIGHT: 18px; LEFT: 0px; TOP: 0px; WIDTH: 580px" tabIndex=1 >

                    </TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
            </TR>
            <TR>
                    <TD ColSpan=7>
        <OBJECT classid=CLSID:1340A251-2064-11D4-861B-9060978DF42E
         codebase="../cab/ucGrid.CAB#version=1,0,0,50" id=ctlarr_Detalle
        style="HEIGHT: 250px; LEFT: 0px; TOP: 0px; WIDTH: 630px" tabIndex=5
        VIEWASTEXT><PARAM NAME="_ExtentX" VALUE="16669"><PARAM NAME="_ExtentY" VALUE="6615"><PARAM NAME="gridCaption"
VALUE="POLIZAS"></OBJECT>
                    </TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
            </TR>
            <TR>
                    <TD ColSpan=6>
                    <TABLE WIDTH="100%" BORDER=0 CELLSPACING=1 CELLPADDING=1>
                            <TR align=right>
                                    <TD align=left>Usuario:
                                    <INPUT id=txtSysUser_Nombre name=txtSysUser_Nombre readonly
                                     value="<%=varSysUser_Nombre%>" style="HEIGHT: 18px; LEFT: 0px; TOP: 0px;
WIDTH: 200px" >
                                    </TD>
                                    <TD>Cargos:
                                    <INPUT id=txtTotalDebe name=txtTotalDebe readonly
                                    value="<%=varContaPol_TotalPol%>"
                                    style="HEIGHT: 18px; LEFT: 0px; TOP: 0px; WIDTH: 100px">

                                    </TD>
                                    <TD>Abonos:
                                    <INPUT id=txtTotalHaber name=txtTotalHaber readonly
                                    value="<%=varContaPol_TotalPol%>"
                                    style="HEIGHT: 18px; LEFT: 0px; TOP: 0px; WIDTH: 100px">

                                    </TD>
                            </TR>
                    </TABLE>
                    </TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
            </TR>

</TABLE>
<%                  Case else 'INSERT   %>
<table valign="Top" bgColor="#336699" border="1" cellPadding="1" cellSpacing="1" width="100%" background =""
borderColorLight="dodgerblue" style="WIDTH: 100%">
    <tr>
      <td>
        <p align="center"><font color="white">Alta de Polizas</font></p>
      </td>
    </tr>
</table>
```

```
<!-- >              Esta tabla es independiente de la de arriba y contiene un registro de dos columnas
          en cada columna esta otra tabla          <-->
<TABLE WIDTH="100%" BORDER=0 CELLSPACING=1 CELLPADDING=1>
          <TR>
                    <TD>Compañía:</TD>
                    <TD ColSpan=5>
                              <INPUT id=txtCia_Desc name=txtCia_Desc
                               Value="<%=varCia_Desc%>" readonly style="HEIGHT: 18px; LEFT: 0px; TOP: 0px; WIDTH:
580px" >
                    </TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD></TD>
                    <TD rowspan=1>
                              <TABLE WIDTH="100%" BORDER=0 CELLSPACING=1 CELLPADDING=1>
                                        <tr>
                                                  <td align="right">
                                                            <IMG height=20 id=btnInsertar name=btnInsertar
onmouseout=LowLight() onmouseover=HighLight() src="../../images/btnInsertar_Low.gif" style="CURSOR: hand" width=80>
                                                  </td>
                                        </tr>
                              </TABLE>
                    </TD>
          </TR>
          <TR>
                    <TD>Foranea:</TD>
                    <TD>
                    <%
                              set objCbo = CreateObject("util_Contabilidad.HTML")
                                        varcboReg_Id = objcbo.CboList ("cboReg_Id
tabIndex=1","15","200","Region","Reg_Id","Reg_Id+' '+Reg_Desc","Where Reg_Status = 1 Order by  Reg_Desc", varReg_Id)
                                        if left(varcboReg_Id,1) <> "<" then
                                                  varReg_Id= left(varcboReg_Id,4)
                                                  Response.Write mid(varcboReg_Id,5)
                                        else
                                                  Response.Write varcboReg_Id
                                        end if
                              set objCbo = nothing
                    %>
                    </TD>
                    <TD>Fraccionamiento:</TD>
                    <TD>
                    <%
                              set objCbo = CreateObject("util_Contabilidad.HTML")
                                        varcboFra_Id = objcbo.CboList ("cboFra_Id
tabIndex=2","15","100","FraCia","Fra_Id","Fra_Id","Where Status = 1 Order by  Fra_Id", varFra_Id)
                                        if left(varcboFra_Id,1) <> "<" then
                                                  varFra_Id= left(varcboFra_Id,4)
                                                  Response.Write mid(varcboFra_Id,5)
                                        else
                                                  Response.Write varcboFra_Id
                                        end if
                              set objCbo = nothing
                    %>
                    </TD>
                    <TD>Fecha:</TD>
                    <TD>
     <OBJECT classid=clsid:20DD1B9E-87C4-11D1-8BE3-0000F8754DA1
     codeBase=../cab/Mscomct2.ocx id=ctlFecha name=ctlFecha tabIndex=3
     style="HEIGHT: 18px; LEFT: 0px; TOP: 0px; WIDTH: 100px" width=100
     VIEWASTEXT><PARAM NAME="_ExtentX" VALUE="2646"><PARAM NAME="_ExtentY" VALUE="476">
     <PARAM NAME="_Version" VALUE="393216"><PARAM NAME="MousePointer" VALUE="0">
     <PARAM NAME="Enabled" VALUE="1"><PARAM NAME="OLEDropMode" VALUE="0">
     <PARAM NAME="CalendarBackColor" VALUE="-2147483643">
     <PARAM NAME="CalendarForeColor" VALUE="-2147483630">
     <PARAM NAME="CalendarTitleBackColor" VALUE="-2147483633">
     <PARAM NAME="CalendarTitleForeColor" VALUE="-2147483630">
     <PARAM NAME="CalendarTrailingForeColor" VALUE="-2147483631">
     <PARAM NAME="CheckBox" VALUE="0"><PARAM NAME="CustomFormat" VALUE="dd/MM/yyyy">
     <PARAM NAME="DateIsNull" VALUE="0"><PARAM NAME="Format" VALUE="662831107">
     <PARAM NAME="UpDown" VALUE="0"><PARAM NAME="CurrentDate" VALUE="36494">
     <PARAM NAME="MaxDate" VALUE="2958465"><PARAM NAME="MinDate" VALUE="-109205">
     </OBJECT>
                    </TD>
                    <TD></TD>
          </TR>
          <TR>
                    <TD>Periodo:</TD>
                    <TD>
                    <INPUT id=txtPeriodo_Id name=txtPeriodo_Id readonly
                    value="<%=varPeriodo_Desc%>"
                    style="HEIGHT: 18px; LEFT: 1px; TOP: 2px; WIDTH: 100px" >

                    </TD>
                    <TD>Documento:</TD>
                    <TD>
                    <%
                              set objCbo = CreateObject("util_Contabilidad.HTML")
                                        varcboTipoPoliza_Id = objcbo.CboList ("cboTipoPoliza_Id
tabIndex=4","15","100","TipoPoliza","TipoPoliza_Id","TipoPoliza_Id + ' ' + TipoPoliza_Desc","Where TipoPoliza_Status =
1 Order by TipoPoliza_Id", varTipoPoliza_Id)
                                        IF left(varcboTipoPoliza_Id,1) <> "<" then
                                                  varTipoPoliza_Id= left(varcboTipoPoliza_Id, 2)
```

```
                                                Response.Write
Mid(varcboTipoPoliza_Id,instr(varcboTipoPoliza_Id,"<"))
                                        else
                                                Response.Write varcboTipoPoliza_Id
                                        end if
                                set objCbo = nothing
                        %>
                        </TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                </TR>
                <TR>
                        <TD>Concepto:</TD>
                        <TD ColSpan=5>
                                <INPUT id=txtContaPol_Concepto name=txtContaPol_Concepto
Value="<%=varContaPol_Concepto%>" style="HEIGHT: 18px; LEFT: 0px; TOP: 0px; WIDTH: 580px" tabIndex=1 >

                        </TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                </TR>
                <TR>
                        <TD ColSpan=7>
        <OBJECT classid=CLSID:1340A251-2064-11D4-861B-0060978DF42E
        codeBase=../cab/ucGrid.CAB#version=1,0,0,50 id=ctlarr_Detalle
        style="HEIGHT: 250px; LEFT: 0px; TOP: 0px; WIDTH: 630px" tabIndex=5
        VIEWASTEXT><PARAM NAME="_ExtentX" VALUE="16669"><PARAM NAME="_ExtentY" VALUE="6615">
        <PARAM NAME="gridCaption" VALUE="POLIZAS"></OBJECT>
                        </TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                </TR>
                <TR>
                        <TD ColSpan=6>
                        <TABLE WIDTH="100%" BORDER=0 CELLSPACING=1 CELLPADDING=1>
                                <TR align=right>
                                        <TD align=left>Usuario:
                                        <INPUT id=txtSysUser_Nombre name=txtSysUser_Nombre readonly
                                         value="<%=varSysUser_Nombre%>" style="HEIGHT: 18px; LEFT: 0px; TOP: 0px;
WIDTH: 200px" >
                                        </TD>
                                        <TD>Cargos:
                                        <INPUT id=txtTotalDebe name=txtTotalDebe readonly style="HEIGHT: 18px; LEFT:
0px; TOP: 0px; WIDTH: 100px">
                                        </TD>
                                        <TD>Abonos:
                                        <INPUT id=txtTotalHaber name=txtTotalHaber readonly style="HEIGHT: 18px; LEFT:
0px; TOP: 0px; WIDTH: 100px">
                                        </TD>
                                </TR>
                        </TABLE>
                        </TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                        <TD></TD>
                </TR>

</TABLE>
<%End Select%>
        <input type="hidden" name="hdnAction" id="hdnAction" Value="<%=varAction%>">
        <input type="hidden" name="hdnarr_Detalle" id="hdnarr_Detalle" Value="<%=vararr_Detalle%>">
</form>

</body>
</html>
<SCRIPT ID=clientEventHandlersVBS LANGUAGE=vbscript>
<!--
Sub window_onload
        document.all("ctlarr_Detalle").gridDetallePoliza = document.all("hdnarr_Detalle").value
End Sub

Sub btnInsertar_onclick()
        dim varPolizaDetalle
        varPolizaDetalle
        =document.all("ctlarr_Detalle").gridDetallePoliza
        if varPolizaDetalle = "ERROR" then
                msgbox  "El detalle de la poliza no es correcto.",vbOkOnly+vbExclamation,"Polizas"
                exit sub
        end if
        document.all("hdnarr_Detalle").value              =varPolizaDetalle
        document.all("hdnAction").value                                   ="INSERTAR"
        document.frmContaPoliza.submit()
End Sub
```

```
Sub btnBorrar_onclick()
        document.frmContaPoliza.submit()
End Sub

Sub btnActualizar_onclick()
        document.frmContaPoliza.submit()
End Sub

Sub ctlarr_Detalle_TotalizaCargoAbono(TotalCargo,TotalAbono)
        document.all("txtTotalDebe").value              =TotalCargo
        document.all("txtTotalHaber").value             =TotalAbono
End Sub
Sub txtContaPol_Concepto_OnBlur
        document.all("ctlarr_Detalle").Concepto = document.all["txtContaPol_Concepto").value
End Sub


-->
</SCRIPT>
```

GLOSARIO

A

ActiveX. Controles propios del entorno Windows, utilizados en programación. Denominados antiguamente OCX (y aún conservan esta extensión como parte del nombre) son módulos separados del programa principal, que se han de distribuir con el mismo. Tienen la ventaja de ser portables entre unos y otros lenguajes y su contenido puede ser cualquiera, no tienen una utilidad específica para un desarrollo concreto. Están basados en la tecnología COM de Microsoft.

Administrador de Bases de Datos. (DBA) La persona encargada del mantenimiento de la base, tanto de los datos, su integridad, manejo del gestor de datos, los índices (por corrupción o cualquier motivo), de directorios y alias, consultas y sentencias desde fuera de programas, igual que conexiones o transacciones, etc., como del acceso de usuarios, autorizaciones, palabras de paso. Depende totalmente de la base de datos de que se trate.

Algoritmo. A partir de un conjunto de datos bien definidos o de un problema concreto, es el conjunto de reglas o de procedimientos lógicos destinados a conseguir el resultado idóneo. Se emplea en programación.

B

Base de Datos. Conjunto de ficheros dedicados a guardar información relacionada entre sí, con referencias entre ellos de manera que se complementen con el principio de no duplicidad de datos. Dependiendo de cómo se vinculen dan lugar a B.D. jerárquicas, relacionales, etc. Un caso especial de éstas son las documentales, que, como su nombre indica, estan diseñadas para almacenar volúmenes grandes de documentos, lo que genera una problemática distinta por los sistemas de búsqueda.

Browser. Visualizador de páginas en Internet, como pueden ser Netscape o I. Explorer.

Buffer. Están relacionados con las memorias del ordenador (sin confundir éstas con la capacidad que puedan tener los discos o dispositivos de grabación similares). Mientras se efectúa un proceso, como un cálculo matemático, una salida por pantalla, por impresora, etc. los resultados intermedios que se van elaborando (en el caso de cálculos, por ej.) o los finales (como los dirigidos a una impresora) son guardados en espacios de memoria especiales, los buffers, y la información que almacenan tiene vigencia hasta que el proceso termina, en que se vacían. Un buffer de impresión puede contener listados mientras se van produciendo, pero una vez los datos se trasladan al papel desaparecen de esa memoria intermedia.

Bug. Termino que se ha puesto de moda para indicar un fallo informático de cualquier tipo.

C

Caché. Es una de las memorias del ordenador, muy rápida, que contiene las operaciones más frecuentes o las últimas realizadas con lo que aumenta considerablemente la velocidad de los procesos al evitar en muchos casos el acceso a memorias más léntas. También se le da este nombre a espacios en disco destinados a contener información reciente obtenida de medios lentos como
Internet, con lo que se evitan tiempos de espera en las cargas de las páginas.

CGI. Common Gateway Interface. Sistema de programación en Internet que permite la generación dinámica de documentos. Esta recibe unas peticiones en forma de "parámetros" desde un servidor y compone un documento en HTML de respuesta que es transmito desde el ordenador cliente. La creación de los programas puede hacerse de forma y con lenguajes muy variados, C, Delphi, etc. y en principio son (o fueron) independientes de la plataforma o el Sistema Operativo

que los albergaba. Hoy existen sistemas más sofisticados para su realización, en entorno Windows se utilizó una especificación Win-CGI y sobre todo otras dos que facilitan muchos trámites al conseguir una comunicación directa entre servidor y cliente que son NSAPI de lo que era Netscape e ISAPI de Microsoft.

Ciclo de vida de un sistema. Desde que se inicia la necesidad de creación de un sistema informático, hasta que se decide que esta obsoleto, los pasos posibles. Como ejemplo, en España, patrocina el Consejo Superior de Informática el llamado Métrica, en él se divide el ciclo en distintas fases: planificación, análisis, diseño, construcción e implementación. Cada una de ellas está subdividida en múltiples subfases o actividades.

Cliente/Servidor. Se le suele llamar así a la arquitectura a dos capas, es decir, una capa servidor, u ordenador que contendrá los datos y los programas gestores asociados, y capas clientes, u ordenadores que se dirigiran al anterior para obtener la información.

Cluster. Unidad de almacenamiento física en discos. Pensando que un disco tiene sectores y pistas básicamente, dependiendo de la cantidad de éstos, el cluster es más grande o menos, por lo que el tamaño de esa unidad es variable. En los primeros disquettes de una cara, un cluster era un sector, en los discos duros nuevos depende de éstos.

COM. Component Object Model Estas tecnologías, originarias de Microsoft, sirven para el diseño de componentes, en realidad de Objetos programables, y la base de OLE tanto en sus variantes OLE DB (DataBase) como OLE DS (Directory Services) y ActiveX.

Comercio electrónico. Es la venta de productos a través de Internet, con sitios especiales que se consideran seguros para hacer los pagos, generalmente por el estandard SSL que funciona por medio de sistemas de criptográfía y servidores de certificación.

D

DBMS. DataBase Management System, Sistema de utilización de Bases de Datos. Ver SGBD.

DCOM. Proviene de COM Distribuido. Diseñado para el acceso a componentes COM a través de redes.

Digital. Señales u ondas cuya información se representa por caracteres numéricos. A nivel informático, por binarios. Sería la alternativa de las analógicas.

Dominio. Hay dos sistemas principalmente de incluir una dirección de páginas Web en Internet, la utilización de espacios generalmente gratuitos y que "cuelgan" de una empresa de suministros o el alta de una propia. Tanto la suministradora de servicios en su momento como el registro de una dirección propia lo que hacen es registrar un dominio, o un nombre registrador en un ordenador al efecto y que asigna un IP propio.

E

E-Mail. Correo electrónico.

Encriptación. Encriptar es hacer ilegible un escrito por medio de aplicar al texto un algoritmo.

Encapsulación. Una de las tres bases de la Programación Orientada a Objetos y que además diferencia lo que es un Objeto de cualquier otro tipo de código, como una función o un procedimiento. Lo que significa, es casi lo que su nombre indica, encapsula o aisla lo que el objeto contiene, de manera que sólo es accesible de unas determinadas formas que se han creado en el mismo, unos parámetros o unos métodos precisos.

F

Firma electrónica. Versión, a nivel de comunicaciones, de la firma digital tradicional.

FTP. File Transfer Protocol. Protocolo para la transferencia de ficheros. Se utilizan servidores

FTP, ya sean particulares o públicos. En ellos queda en depósito lo que se haya de transmitir, a diferencia del envío mediante correo electrónico en el que la información se dirige al receptor.
Fuente. Lenguaje fuente es el que utiliza el programador, con la sintaxis que corresponda y perfectamente legible. A partir de él, mediante los compiladores, salvo en lenguajes de programación concretos, se genera el código que el ordenador es capaz de entender.

G

Gateway. Ver Pasarela.

GUI. Graphic User Interface. Proporciona el cambio de las pantallas conocidas como "de modo texto" a las gráficas, con manejo de puntos en lugar de caracteres.

H

Hacker. Término denostado por los medios de comunicación que se refiere al informático especializado o con inquietudes de salvar determinados retos complejos. Popularmente se le considera dedicado a la infiltración en sistemas informáticos con fines destructivos.
Herencia. Termino utilizado en Programación Orienta a Objetos. Teniendo un objeto con unas características concretas (ascendiente), se podría definir como herencia a la derivación de éste en otro nuevo que incluye sus características mas las que se incluyan (descendiente) Hay una herencia simple. Imagínate un diagrama en forma de árbol, de una clase pueden descender un montón de otras clases, pero lo que normalmente se ve es que una clase siempre parte de otra, o que de una parta distintas o de la originaria partan todas las demás. Estaríamos en herencia simple, de manera que al generar un componente nuevo tienes absolutamente claro de qué otro desciende. Pero hay lenguajes, como ocurre en C++, que esto puede no ser así, puede haber una herencia múltiple cuando a una clase le pueden corresponder distintos ancestros. ¿Que ocurre con ello? pues resulta que no hereda de un solo "padre", sino de múltiples, por lo que puede tener propiedades tanto de unos como de otros.
Hipertexto. Es una de las características de las páginas de Internet. Se le denomina así a la capacidad de saltar de un documento a otro por medio de imágenes o de "puntos calientes" en el propio texto con solo pulsar la tecla del ratón sobre él, lo que permite "navegar" ya sea dentro de una Web o hacia otras. Se pueden resaltar de muchas formas, en estas páginas en concreto están en
negrita y subrayados. Ver también Enlaces.
Host. Ver Servidor.
HTML. Hypertext Markup Language. Al redactar un escrito en un procesador de textos normal, lo que se ve en pantalla no es lo que realmente se graba en el disco. Si comprobásemos el trasfondo de lo escrito, aparecerían caracteres por todas partes ilegibles, pero que sirven al procesador para crear la apariencia de lo que se ve. Cuando hablamos de páginas Web ocurre mas o menos igual. Tras
la apariencia e ésta misma página se esconde la realidad de lo que hay escrito, su color, fondo, tipo de letra, etc. Esto se consigue a través del lenguaje HTML.
HTTP. Es el protocolo o las reglas de funcionamiento de los servidores WWW, que son los encargados de mantener este tipo de páginas.

I

Indexar. Que también se dice. Teniendo un fichero de datos, es la creación de otro archivo paralelo que conseguirá, por medio de algoritmos, sistemas de ordenación y búsquedas directas

(o las que se conocían por secuenciales indexadas) de manera que lo que guardan es el término (campo) por el que se realiza el proceso de indexación y la dirección física donde se localiza en el fichero origen. El resultado es lo que se conoce como fichero índice.

Interfaz. Este término se utiliza con distintas acepciones,. Principalmente es un lugar físico común entre dos dispositivos informáticos y que permite la conexión entre ellos. No obstante se habla de interfaz gráfica, de usuario, etc. y no tiene una relación con lo explicado. Interfaz de usuario. Es la manera de funcionar el ordenador de cara al usuario, o mejor, la relación de ambos, es decir, cómo responde a los sucesos o acciones.

Internet. Seguro que es el término más popular en estos momentos cada vez que se habla de informática en general. Se le pueden dar definiciones puntuales, como "red de redes" que aunque no nos sirve para entender nada, está bien, pues en definitiva no es más que redes de ordenadores interconectadas. Como va más allá de esta realidad, vamos a comentarlo con algo más de detenimiento. Históricamente tiene su origen en ARPANET, proyecto militar estadounidense, en 1.969. A partir de los años 80 se extiende al mundo científico y en la actualidad es universal. Uno de los puntos importantes es el uso de un protocolo propio, TCP-IP, donde IP son las siglas de Protocolo de Internet, y también se denominan así a las direcciones de los servidores, son unas series de números que los identifican y que, para comodidad del usuario, se traducen a nombres, como cualquiera que se puede ver en una página de enlaces o ésta misma donde nos encontramos. A estos "apodos" se les denomina direcciones DNS. Otro de los aspectos que han influido en la universalización de Internet es el sistema WWW o Web, tanto es así que practicamente se identifica uno con el otro a nivel popular. Y ¿para qué sirve Internet?. Pues el primer punto y esencial es la obtención de información, el medio actual más difundido es la Web, aunque se van desarrollando sistemas sobre ella para transacciones de todo tipo, el problema es que el volumen de la red es de tal magnitud que a pesar de los "buscadores" o programas especializados en localizar datos, hay que saber muy bien qué se busca y cómo hacerlo. El segundo en importancia es el correo electrónico, que permite el envío de mensajes entre usuarios de la red en cualquier lugar del mundo. La transferencia de ficheros o FTP, y bastantes más servicios que no es cuestión de detallar aquí. Solo mencionar, que los sistemas de conferencia (diálogo hablado a través de la red) y la videoconferencia (voz e imágenes) se va implantando paulatinamente.

ISAPI. Ver CGI.

J

Java. Lenguaje de programación derivado del C++, aunque simplificado en principio y adaptado para el funcionamiento en entornos de IntraNet-Internet.

L

LAN. Ver Red de área local.

Linux. Sistema Operativo de reciente aparición y muy rápido ascenso, basado en UNIX, utilizado tanto en microordenadores como en minis, que nace de los proyectos Open Source y parece un claro sustituto de Windows.

Login. Es un identificador y depende del contexto: Puede ser la identificación o confirmación de la palabra de paso. Cualquier programa, sitio, terminal de red, lo que sea, puede tener su palabra de paso como identificativo y su login como validación de este. También hay quien se refiere a él como al conjunto de claves.

M

Mainframe. Se le denomina así, dentro de la división de los ordenadores por su potencia, a los de mayor embergadura.
MDI. Sistemas de funcionamiento o de programación de ventanas bajo entornos Windows. Las siglas provienen de "Interfaz de Documentos Múltiples". Se caracteriza por la dependencia jerárquica entre las ventanas o por una relación de ventanas madre e hijas, el funcionamiento de ellas está vinculado.
Multitarea. Realización simultánea de varios procesos en un ordenador, teniendo en cuenta que éste y su Sistema Operativo lo permitan.
Multiusuario. Utilización simultanea de un ordenador por varios usuarios, o dicho de otra manera, ordenador y Sistema Operativo con capacidad para permitir el trabajar desde distintas consolas con la misma CPU.

N

Navegador. En el ámbito de Internet, es un software capaz de visualizar la información en formato de Web, es decir, pueden utilizar hipertexto y los protocolos propios de Internet, de manera que no sólo se ciñen a páginas en html, sino que igual manejan FTP, que SMTP, grupos de noticias, ayudados por otros softwares pueden manejar cualquier tipo de multimedia, etc.
NFS. Network File System. Sistema creado por Sun Microsystem para compartir ficheros dentro de una red con sistema operativo UNIX.

O

Objeto. Término base de la Programación Orientada a Objetos, en pura teoría podríamos pensar en lo que el hombre conoce, una mesa, un mechero, son cosas que nuestra inteligencia sabe comprender y explicar porque tienen unas características comunes, a pesar de que las diferencien muchas otras, pues hay formas y modelos muy dispares. A nivel informático es similar, un objeto es una parte de código con unas peculiaridades, de manera que podemos crear el objeto "subMesa", que contendrá las básicas de su ancestro pero modificadas, es decir, todo lo que puede hacer "Mesa" con su código, llamadas a funciones, punteros, etc., mas las particularidades que proporcionemos a este "hijo" suyo.
ODBC. Open Data Base Conectivity. Ha sido la base de Windows en sistemas abiertos, es decir que permiten una conectividad entre distintos lenguajes de programación con distintas bases de datos.
OLE. Object Linking and Embedding. Es un concepto complejo del entorno de programación en Windows y que abarca campos muy amplios. Podría decirse que son conjuntos de librerías, que permiten crear conexiones entre aplicaciones distintas, como dirigirse desde un programa determinado a una hoja de cálculo o un procesador de textos. Pero así expresado es muy rudimentario, porque también permite la interacción entre el Sistema Operativo y otros programas, documentos compuestos que alberguen formatos muy distintos desde utilidades y aplicaciones muy variadas. Además, al no ser un estándar único, sino un conjunto, está en continua evolución inclusive en Sistemas Operativos distintos a Windows. Ver COM.

P

Páginas Web. Forma de denominar a las hojas creadas con html que se manejan dentro del entorno WWW.

Pasarela. Gateway. Sirven para enlazar redes que tienen un protocolo distinto, permitiendo la comunicación entre ellas. Aunque hay diferentes dispositivos para enlaces y conexiones, las pasarelas son las de nivel superior en el modelo OSI (a partir del nivel 4), al realizar también funciones de traducción.
Portal. Se refiere a un sitio en Internet o un sitio Web donde se "da entrada" (supongo que por eso el nombre de portal) a distintos servicios. Tradicionalmente servicios de búsqueda y enlace con páginas Webs, correos electrónicos gratuitos o no, información de distinto tipo, etc. Depende cual se trate y la imaginación de sus creadores.
Programa. Instrucciones que varían según el lenguaje que se utiliza, pero cuyo fin es el de controlar las acciones que tiene que llevar a cabo el ordenador y sus periféricos.
Programación Orientada a Objetos (POO). Es un sistema de programación, procedente de la evolución de la programación estructurada, cuyos pilares son la reutilización del código y la facilidad (teórica) de simplificarle. Su base es lo que se denomina Objeto. Protocolo. Es un término de comunicaciones y su función es fijar unas reglas de funcionamiento, a todos los niveles, a las que han de atenerse los distintos sistemas informáticos para poder comprenderse.

Q

Query. Consulta a una base de datos generalmente utilizando sentencias SQL.

S

Sistemas de Información. Se debe considerar un sistema de computación e información como el conjunto de componentes físicos (hardware), lógicos (software), de comunicación (bien redes de cualquier tipo o tipo Internet) y medios humanos (lo que ahora llaman orgware), todo ello unido permite el tratamiento de la información.
Sistemas de Encriptación o Cifrado. La criptografía es un área gigante, porque su origen es muy antiguo y sistemas de cifrado hay muchos, pero existe una división básica: Criptografía de Clave Privada, legible tan sólo por el destinatario que conoce la forma de descifrarlo, a diferencia de la Criptografía de Clave Pública, que puede serlo por distintos destinatarios, ya que en realidad son dos claves y una persona (organismo) certificadora. Cualquier sistema de encriptación o cifrado es un sistema matemático.
Sistema Operativo. Es un conjunto de programas que, se podría decir, sirve de enlace entre el ordenador y el programador usuario. Son los responsables de gestionar los recursos del ordenador, discos duros, memorias, control de periféricos como pantallas, teclados, etc. Se podría decir que unifican y estandarizan el funcionamiento de los ordenadores. Como ejemplo, sin ellos una aplicación tipo contable o cualquier otra, solo tendría funcionamiento en ordenadores de exactas características, gracias a ellos esa misma aplicación será posible utilizarla en cualquiera que comparta el mismo Sistema Operativo.
SMTP. Simple Mail Transfer Protocol. Es el correo electrónico estándar utilizado por Internet.
SNMP. Simple Network Management Protocol. Protocolos para la gestión de redes basadas en TCP/IP.
SQL. Structured Query Language. Lenguaje o sentencias para el manejo y consulta de bases de datos.
SSL. Se ha convertido en el estandard de comercio electrónico en cuanto a protocolo dirigido a la creación de servidores seguros (en realidad el protocolo es el HTTPS, pues se puede utilizar el sistema SSL con otros fines, aunque este sea el generalizado). Utiliza un sistema combinado de claves simétricas y asimétricas (la misma sirve para crear el cifrado y descifrado, o son distintas), a través de una pública, otra privada y una entidad certificadora.

T

TCP-IP. Transmision Control Protocol-Internet Protocol. Protocolo en el que se basa Internet y que en realidad consiste en dos. El TCP, especializado en fragmentar y recomponer paquetes, e IP para direccionarlos hasta su destino.

U

UNIX. Sistema Operativo desarrollado por AT&T en 1969 en lenguaje ensamblador, posteriormente se reescribió a lenguaje C lo que ha hecho posible su desarrollo actual. Es quizás el S.O. más extendido entre ordenadores de tamaño medio. De él han surgido distintos sistemas derivados como lo fue Xinux o el actual Linux. De todas las versiones la que ha tenido mayor implantación es la conocida como System V. Es multitarea y multiusuario.
URL. Uniform Resource Locator. Se conoce por este nombre a las direcciones dentro de Internet, normalmente, aunque no necesariamente, refiriéndonos a páginas Web. En este caso se distinguen por iniciarse con http:// No obstante es una simplificación para el usuario el referenciarlas de esta forma, en realidad son secuencias de números que se dirigen de forma inequívoca a una dirección.

W

Web. Por éste término se suele conocer a WWW (World Wide Web), creado por el Centro Europeo de Investigación Nuclear como un sistema de intercambio de información y que Internet ha estandarizado. Supone un medio cómodo y elegante, basado en multimedia e hipertexto, para publicar información en la red. Inicial y básicamente se compone del protocolo http y del lenguaje html. Un ejemplo claro de páginas de éste tipo, es la que tienes delante en estos momentos.
Windows. Sistema operativo de 32 bits de Microsoft.

BIBLIOGRAFÍA

Madden, Jeff,; Nolan Sean, *Microsoft SQL Server 7.0 Database Training Kit*, Microsoft Press, USA 1999.

Sledge, Orryn,; Spenik, Mark,; *SQL Server DBA Survival Guide*, Sams Publishing, USA 1996.

Lhotka, Rockford, *Professional Visual Basic 6 Distributed Objects Implementing Distributed Objects with VB, MTS, MSMQ and RDS*, Wrox Press LTD, Canada 1999.

Lhotka, Rockford, *Visual Basic 6 Business Objects Enterprise Design and Implementation*, Wrox Press LTD, Canada 1998.

Bortniker, Matthew,;Conrad, M. James, *Professional Visual Basic 6 MTS Programming VB COM for the Distributed Environment*, Wrox Press LTD, Canada 1999.

Pattison, Ted,; Box, Don, *Programming Distributed Applications with COM and Microsoft Visual Basic 6*, Microsoft Press, USA 1998.

Appleman, Dan, *Building COM/Activex Components with Visual Basic 6*, Prenticce Hall, Madrid 2000.

Rogerson, Dale, *Inside COM*, Microsoft Press, USA 1997.

Chappell, David, *Understanding Activex and OLE*, Microsoft Press, USA 1996.

Sturm, Jake, *VB6 UML Design and Development*, Wrox Press LTD, Canada 1999.

Miller, Scott,; Mezick, Daniel,; *Programming Active Server Pages*, Microsoft Press, USA 1997.

Scott, Isaacs, *Inside Dynamic HTML*, Microsoft Press, USA 1997.

Thurrott, Paul,; Cox, Ken,; Banick, Steven,; Fino, Brian M.,; Kindred, James,; *Unleashed Visual Interdev 6*, Sams Publishing, USA 1999.

Banick, Steve,; Morrison, Michael, *Using Visual Interdev 6*, Que Corporation, USA 1998.

Evans, Nicholas D.,; Miller, Ken,; Spencer Ken, *Programming Microsoft Visual Interdev 6.0*, Microsoft Press, USA 1999.

Manheim, Seth,; Morey, Jim, *Microsoft Internet Information Server Resource Kit*, Microsoft Press, USA 1998.

Bramble, David,; Thues, Steve, *Microsoft Internet Information Server 4.0 Training*, Microsoft Press, Usa 1998.

Greer, Tyson, *Understanding Intranets*, Microsoft Press, USA 1998.

Savola, Tom,; Westenbroek, Alan,; Heck, Joseph, *Using HTML Special Edition*, Que Corporation, USA 1995.

Afergan, Michael,; Darnell, Rick,; Farrar, Brian,; Jacobs, Russ,; Medinets, David,; Mulen, Robert,; Ó Foghlú, Micheál, *Web Programming Desktop Reference 6 in 1*,Que Corporation, USA 1996.

Heywood, Drew, *Inside Windows NT server*, New Riders Publishing, USA 1995.

Microsoft Professional Editions, *Microsoft Windows NT Server Resource Kit*, Microsoft Press, USA 1996.

Microsoft Professional Editions, *Microsoft TCP/IP Training*, Microsoft Press, USA 1997.

SITIOS WEB

http://www.microsoft.com/technet
http://msdn.microsoft.com
http://www.microsoft.com/dna
http://www.sun.com
http://www.netscape.com
http://www.oracle.com
http://otn.oracle.com